UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola Superior d'Enginyeries Industrial, Aeroespacial i Audiovisual de Terrassa

Titulació:

Grado en Ingeniería Electrónica Industrial y Automática

Alumne:

Miquel Egea Jiménez

Enunciat TFG:

Desarrollo de un prototipo portátil de electrocardiograma

Director del TFG: Raúl Fernández García

Convocatoria de Lliurament: Primavera (QP 2019)

Abstract

The initial main purpose in this project was to design a portable prototype capable of monitor the electrocardiogram of a human person using three electrodes. This prototype would include the ADS1292R chip and its conditioning electronic circuit. It would transfer the data by wireless to a PC or mobile device.

However, the main objective has not been fulfilled instead, it has been developed an interface using LabVIEW to monitor the electrocardiogram. The ADS1292R microcontroller is included in a shield to be used together with Arduino UNO. The communication between LabVIEW and Arduino is set up by Serial Port.

Agradecimientos

A mi familia: mi madre, mi padre que en paz descanse y mi hermana, por darme siempre la fuerza y el apoyo para seguir adelante ante cada dificultad.

A mi pareja, por estar a mi lado en todas las situaciones difíciles que he atravesado.

A mi tutor, por todos los consejos que me ha sabido ofrecer y todos los problemas que me ha ayudado a resolver a lo largo de estos meses de proyecto.

Índice

Lista de Figuras

Lista de Tablas

1. Introducción

En este proyecto se llevará a cabo el desarrollo de una interfaz para monitorizar el electrocardiograma de una persona.

Para la adquisición de los datos se utilizará la placa microcontroladora Arduino UNO y el chip ADS1292R de Texas Instruments. Este chip será el encargado de convertir los impulsos eléctricos captados por los electrodos a valores digitales. Viene incorporado en el shield ADS1292R distribuido por Protocentral.

Este shield se inserta sobre los pines del Arduino para comunicarse entre sí. La comunicación posterior entre Arduino y PC se realizará mediante el puerto serie, por el momento.

La interfaz estará desarrollada mediante el software LabVIEW, muy utilizado en este tipo de situaciones. En ella, se permitirá visualizar el ECG y la respiración del usuario, así como exportar estos datos a otros formatos de archivo.

Los objetivos iniciales del proyecto fueron realizar un dispositivo portátil capaz de sustituir el shield por un circuito electrónico que, incluyese el chip ADS1292R además de, una serie de componentes electrónicos para acondicionar el circuito y transmitir los datos.

Esta transmisión se realizaría de forma inalámbrica mediante tecnología Bluetooth al destinatario que, en principio sería también un ordenador.

No obstante, debido a varios factores sobre todo personales, el proyecto ha visto limitado su alcance en gran medida:

1. El circuito no abandonara la forma de shield ni la placa microcontroladora de Arduino.
2. La transmisión será mediante el puerto serie.

La distribución de las tareas se puede observar en la Fig. 1.

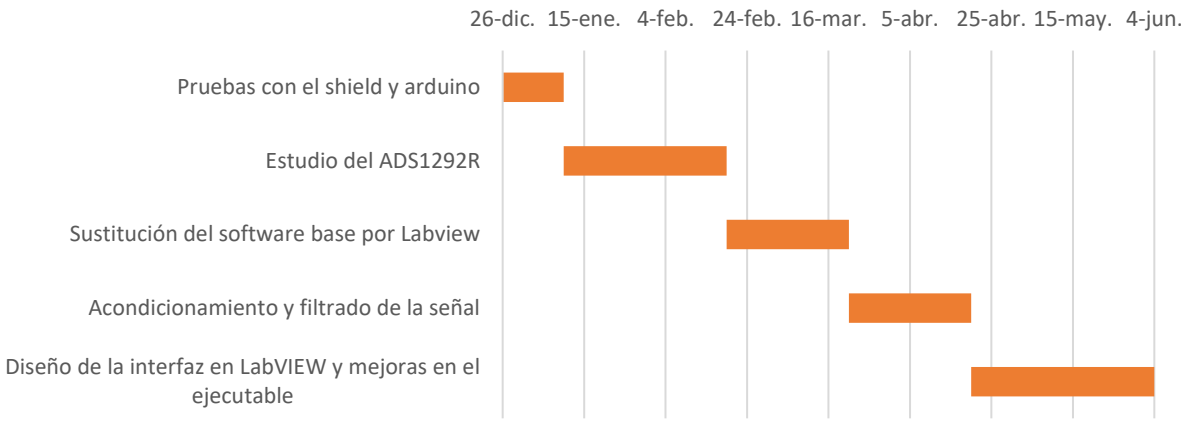


Fig. 1. Diagrama de Gantt del proyecto

El estudio del ADS1292R ocupó más tiempo de lo esperado debido a varios acontecimientos externos al proyecto y por la gran cantidad de información que se debía comprender.

El resto de las tareas fueron sobre los plazos previstos.

1.1. Motivación

La motivación principal de este proyecto se encuentra en mi dedicación durante muchos años al deporte, concretamente al atletismo.

Me propuse inicialmente, como se ha comentado anteriormente, el diseño de un prototipo portable capaz de evaluar la salud del corazón mediante el electrocardiograma. La idea ideal fue transmitir los datos a una aplicación diseñada para smartphones, para la máxima portabilidad.

Este dispositivo se podría utilizar durante los entrenamientos y obtener información de ellos como por ejemplo las pulsaciones por minuto o el ritmo respiratorio.

1.2. Objetivos y alcance del proyecto

La correcta visualización del ECG es sin duda, el objetivo principal del proyecto, así como el diseño de una interfaz clara y limpia en el software de LabVIEW.

Como se ha comentado antes, por el momento la comunicación entre hardware y software será mediante puerto serie. En una futura implementación se utilizará una comunicación inalámbrica mediante Bluetooth.

Para la realización del proyecto es necesario comprender y realizar:

1. El funcionamiento del chip ADS1292R
2. La conversión de los valores a la salida del chip
3. La conexión de los electrodos en el cuerpo
4. La programación de la interfaz en LabVIEW.

2. Fundamentos teóricos

2.1. El corazón y el ciclo cardíaco:

El corazón es uno de los elementos centrales del cuerpo humano. Se puede dividir en dos zonas y en cada una de ellas se encuentra una aurícula y un ventrículo.

Como se puede observar en la Fig. 2 , estas dos cavidades están separadas por una válvula llamada mitral en el lado izquierdo y tricúspide en el lado derecho.

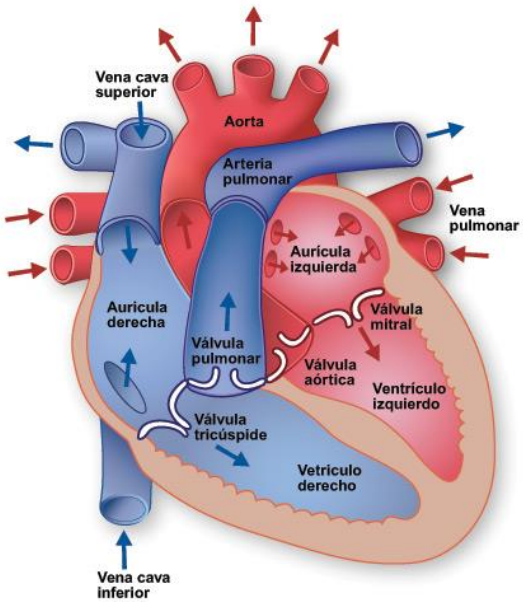


Fig. 2. El corazón y sus zonas más diferenciadas des de una perspectiva frontal

Durante lo que conocemos como un ciclo cardíaco, la sangre entra en ambas aurículas proveniente de las venas cava y pulmonar (fase diástole[1]).[1] A continuación, debido a la presión sobre las válvulas, éstas se abren permitiendo el paso de la sangre a los ventrículos.

[1] Movimiento de relajación y expansión de las arterias que se produce cuando la sangre purificada entra en ellas.

Una vez llenos los ventrículos, se contraen y se produce la apertura de las válvulas aórtica y pulmonar, impulsando la sangre a través de las arterias aorta y pulmonar hacía el resto del cuerpo (fase sístole[2]).

El corazón se relaja y se produce un nuevo ciclo cardíaco. Cada una de estas etapas es provocada por un impulso eléctrico determinado. Estos impulsos se pueden medir y registrar en lo que conocemos como electrocardiograma.

2.2. El electrocardiograma:

El electrocardiograma es una prueba médica que registra la actividad eléctrica del corazón, concretamente del miocardio (el tejido muscular de éste). Estos impulsos eléctricos están comprendidos en un rango de frecuencias de 0,5 Hz a 150 Hz y pueden tener una magnitud de entre 0,1mV a 10mV [2].

En este proyecto se tendrán en cuenta el rango de frecuencias de 0,5 Hz a 40 Hz ya que posee los datos más básicos para su representación.

La representación del ECG tiene un aspecto como el de la Fig. 3.

Fig. 3 Aspecto de un ECG

Se realiza sobre papel milimetrado en el que cada división mayor representa 0,02 segundos en la dirección de las X y 0,5 mV en la dirección de las Y.

Para llevar a cabo esta prueba, se deben colocar los electrodos en diferentes puntos del cuerpo. Existen unos puntos de colocación estándares que se conocen como derivaciones.

[2] Movimiento de contracción de las arterias para empujar la sangre que contienen.

2.2.1. Las derivaciones

Las derivaciones son las mediciones de la diferencia de potencial eléctrico entre dos electrodos colocados sobre la piel.

Las derivaciones más comunes son las bipolares, conocidas como periféricas. Estas utilizan la diferencia de potencial entre las extremidades superiores e inferior izquierda como podemos observar en la **¡Error! No se encuentra el origen de la referencia.**.

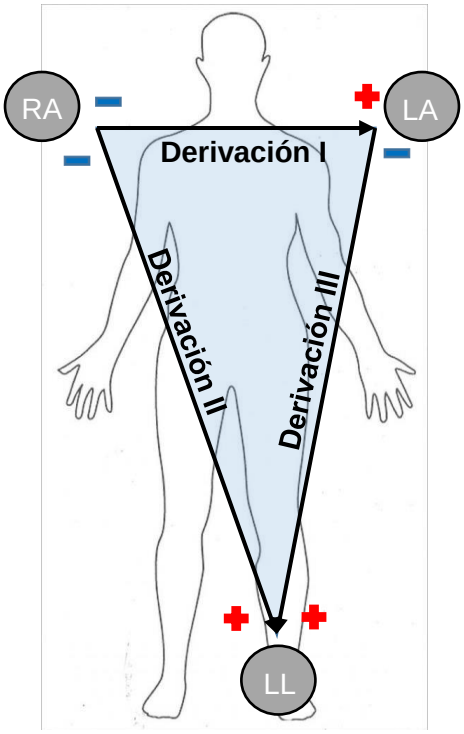


Fig. 4 Derivaciones bipolares

Concretamente, cada derivación determina:

- Derivación I:

  Determina la diferencia de potencial entre el brazo derecho (RA) y el brazo izquierdo (LA). Se considera el brazo izquierdo como positivo.

- Derivación II:

  Determina la diferencia de potencial entre el brazo izquierdo (LA) y la pierna izquierda (LL). Se considera la pierna izquierda como positivo.

- Derivación III:

  Determina la diferencia de potencial entre el brazo derecho (RA) y la pierna izquierda (LL). Se considera la pierna izquierda como positivo.

Esta representación de la Fig. 4, se conoce como triangulo de Einthoven y nos permite comprobar si se está realizando correctamente el electrocardiograma.

De acuerdo con la ley de Einthoven, DII = DI + DIII. Si en algún momento la suma de los potenciales de las derivaciones I y III no equivalen al potencial de la derivación II, los electrodos están mal colocados y por lo tanto no se está representando correctamente el ECG.

En este proyecto se utilizarán las tres derivaciones bipolares.

2.2.2. Caracterización del ECG

Durante cada ciclo cardíaco se registran unas ondas y segmentos que se pueden observar en la Fig. 5.

Como se ha comentado en el apartado anterior, cada instante de un ciclo es producido por un impulso eléctrico determinado. Las ondas y los segmentos reflejados en el ECG nos dan información sobre cada momento del ciclo.

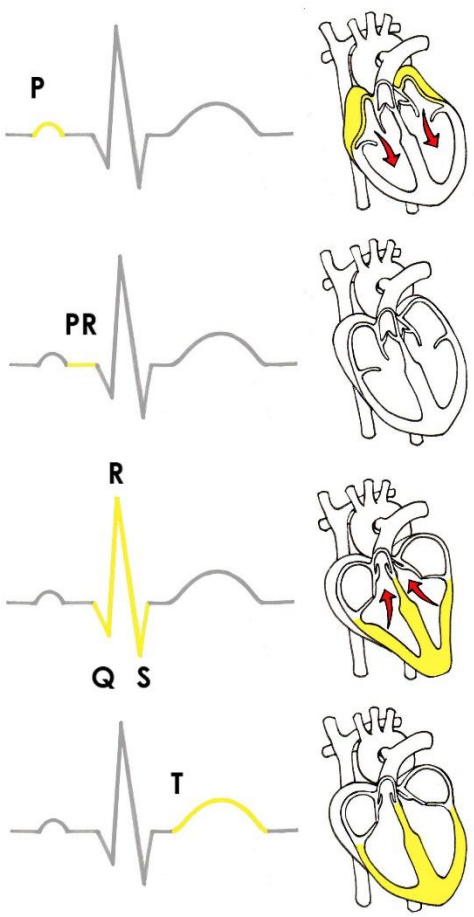


Fig. 5. Los impulsos eléctricos según el momento del ciclo cardíaco

El primer elemento que se produce en un ciclo cardíaco es la onda P. Corresponde a la entrada de la sangre proveniente de las venas a las aurículas en el momento de su despolarización. Está compuesta por la superposición de la actividad eléctrica de las aurículas de ambos lados. La duración normal de una onda P es menor a 0,12 segundos y con una tensión máxima de 0,25 mV.

La Fig. 6 representa una onda P:

1 - Normal.

2 - Con la aurícula derecha dilatada.

3 - Con la aurícula izquierda dilatada.

4 - Con ambas aurículas dilatadas.

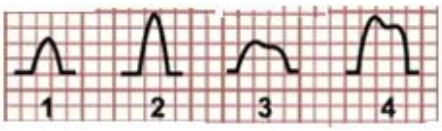


Fig. 6. Alteraciones en la onda P

A continuación, se produce un instante sin ninguna onda visible en el ECG. Se conoce como segmento PR y representa el momento en el que la sangre llena los ventrículos hasta ejercer una presión determinada sobre las válvulas. Es un segmento normalmente isoeléctrico[3] y se produce entre el final de la onda P y el inicio del complejo QRS. La duración de estos dos fenómenos se conoce como intervalo PR. La duración normal del intervalo PR es de entre 0,12 y 0,2 s.[3]

Una vez ha concluido el intervalo PR, aparece el complejo QRS que corresponde a la contracción de ambos ventrículos, es decir su despolarización, para impulsar la sangre hacia las arterias. Normalmente ambos lados se activan a la vez, suele influir en mayor medida la despolarización del ventrículo izquierdo que la del derecho. Esto es debido a que el izquierdo tiene mayor masa muscular. El complejo QRS está formado por tres ondas consecutivas (Q, R y S). La primera es la onda Q y es negativa (aparece de forma descendente). A continuación, encontramos la onda R que es positiva y finalmente aparece la onda negativa llamada S. La duración del complejo QRS oscila entre 0,05 y 0,1 segundos. Durante el complejo QRS, también ocurre la repolarización de las aurículas,

[3] Sin tensión eléctrica

pero su intensidad es tan inferior a la despolarización de los ventrículos que no se aprecia en la representación del ECG[3].

El intervalo comprendido entre el final del complejo QRS y el inicio de la siguiente onda, se conoce como segmento ST. Este segmento es también isoeléctrico, en condiciones normales. En este momento las aurículas ya han vuelto al reposo mientras que los ventrículos todavía no se han repolarizado, es decir siguen en estado de contracción. La duración de este segmento ocupa un tiempo de unos 0,12 s.

En el momento en el que se repolarizan los ventrículos, se produce la onda T. Generalmente es de menor amplitud que el complejo QRS que le precede.[4] En un electrocardiograma normal, la onda T es positiva y debe ir acorde con la dirección del complejo QRS. Es decir, si el QRS es positivo en una derivación la onda T debe serlo también en esa derivación.

La duración entre el inicio de la onda Q y el final de la onda T se conoce como intervalo QT. Indica el tiempo total de contracción de los ventrículos. Normalmente dura unos 0,4 s.

En la Tabla 1, se puede observar un resumen de cada elemento, sus características y el fenómeno que los genera:

Elemento	**Duración normal**	**Fenómeno**
Onda P	Hasta 0,12 s	Despolarización de las aurículas
Segmento PR	Hasta 0,08 s	Tiempo entre el fin de la despolarización de las aurículas y el inicio de la despolarización ventricular
Complejo QRS	Hasta 0,1 s	Despolarización ventricular y repolarización auricular.
Intervalo QT	Unos 0,4 s	Tiempo de despolarización y repolarización ventricular.

Tabla 1 Resumen de los fenómenos cardíacos

3. Estado del arte

3.1. Normativas

Existen normativas referentes a la seguridad, el rendimiento y la compatibilidad electromagnética de los equipos y sistemas eléctricos médicos.

Están recogidas en la familia de normas IEC 60601 y contiene unos 60 estándares particulares, aprobados por la IEC (Comisión Internacional de Electrotecnia).

En España esta familia de normativas está recogida por la UNE-EN-60601.

3.2. Propuestas actuales

Actualmente se están implementando dispositivos que son totalmente portátiles y autónomos. La mayoría de los productos son compatibles con smartphones y utilizan aplicaciones que reciben los datos vía Bluetooth.

1. QARDIOCORE

Este dispositivo, como se puede observar en la Fig. 7 y Fig. 8, está formado por dos bandas que se colocan sobre el pecho y mediante Bluetooth se visualiza el ECG en la aplicación móvil.

Algunas de las características de este dispositivo son:

- Respuesta en frecuencia entre 0,05 Hz y 40 Hz
- 600 muestras por segundo
- Resolución de muestreo: 16 bits
- Bluetooth 4.0
- Precio (oficial): 499€

Fig. 8 Bandas detectoras del QARDIOCORE

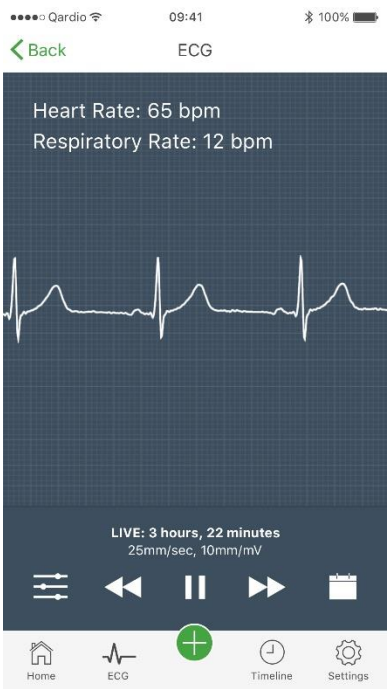


Fig. 7 Aplicación para smartphones en el QARDIOCORE

2. KARDIA

Esta banda del tamaño de una tarjeta de crédito, como se puede observar en la Fig. 9, juntamente con la aplicación para smartphones o, en este caso también, para relojes inteligentes (de momento solamente Apple Watch) registra el ECG en 30 segundos y lo transmite al dispositivo.

Está distribuido por la marca AliveCor por un precio de 149 €.

Mide los impulsos a partir de los dedos colocados en las almohadillas y está certificado para uso pediátrico en Europa.

Fig. 9 Banda Kardia para la detección del ECG

3.3. Materiales

3.3.1. Shield ADS1292R

El shield ADS1292R como se puede observar en la Fig. 10, es una placa microcontroladora principalmente compuesta por el convertidor analógico digital ADS1292R.

Fig. 10 Shield ADS1292R

Este chip nos permite muestrear los impulsos eléctricos del ECG y la respiración, convertirlos a valores digitales y transmitirlos mediante el puerto serie para su posterior tratamiento.

Los tres electrodos que se usarán para captar estas señales se conectan utilizando el puerto Jack 3.5. En la Fig. 11 se puede observar los electrodos.

Fig. 11 Los tres electrodos incluidos en el shield ADS1292R

Además, incluye 10 parches desechables para la conexión de estos tres electrodos en el cuerpo del usuario.

3.3.1.1. Estructura y características.

En la Fig. 12 se muestra el diagrama de bloques del chip ADS1292R. Dispone de dos canales de conversión de datos. Uno se utilizará para los valores del ECG y el otro para los valores de la respiración.

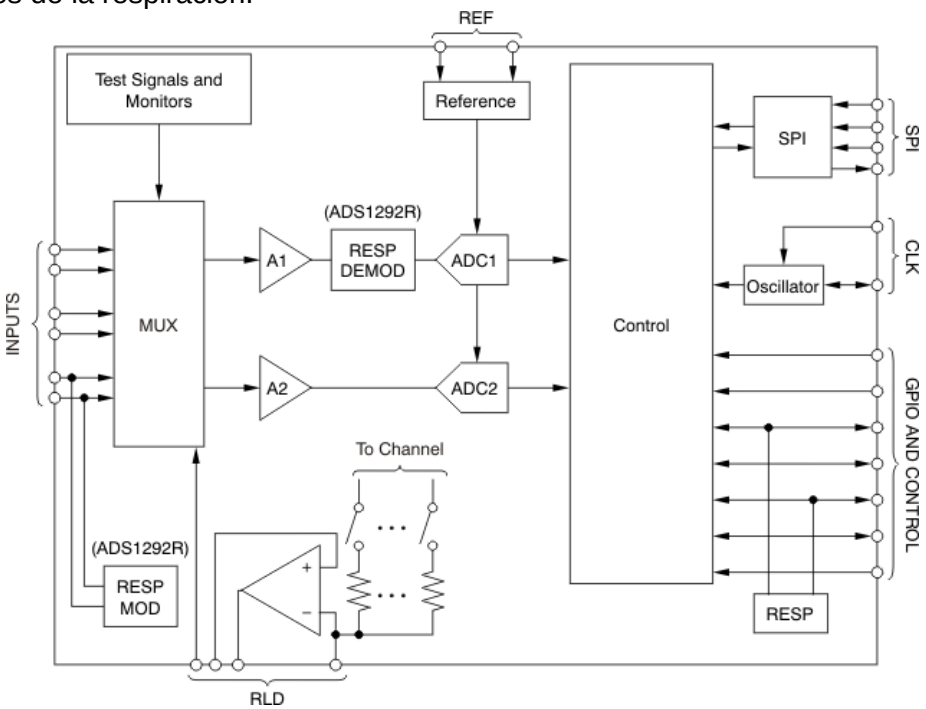


Fig. 12 Esquema de bloques del ADS1292R

En la Fig. 13 se muestran los pines de conexión del chip ADS1292R.

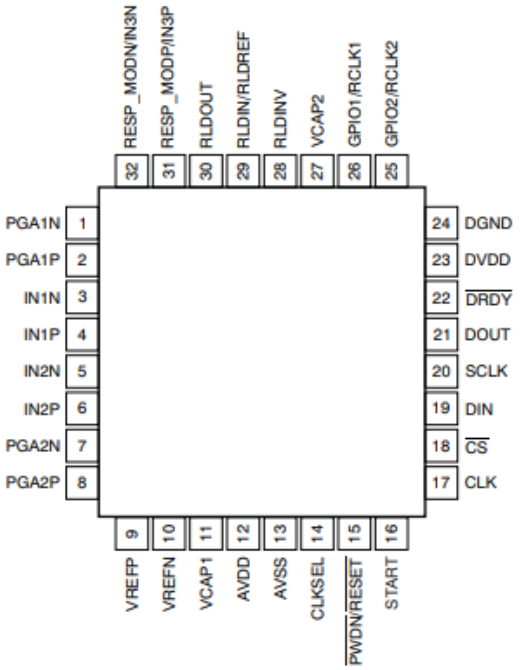


Fig. 13 Esquema de los pines del chip ADS1292R

En la Tabla 2 se resumen las principales características eléctricas del chip ADS1292R.

PARÁMETRO	VALORES
DC Input Resistance	1000 MΩ (mínimo)
Bandwidth	8.5 kHz (típico)
ADC Resolution	24 bits
Date Rate	125 – 8.000 (SPS)
Common Mode Rejection Ratio (CMRR)	-105 dB (mínimo)
Total Harmonic Distortion (THD)	-82 dB (típico)
Analog Supply (AVDD)	2.7 – 5.25 V
Digital Supply (DVDD)	1.7 – 3.6 V
Programmable Gain	1, 2, 3, 4, 6, 8, 12

Tabla 2 Principales características eléctricas del chip ADS1292R

Está alimentado mediante 3V. Sin embargo, la entrada del chip está configurada en modo diferencial por lo que el rango de tensiones de entrada va desde -V_{REF} hasta + V_{REF}:

Rango = $(\pm \text{VREF})/Ganancia = (2 \times \text{VREF})/Ganancia$

Para programar la ganancia, cada canal dispone de un PGA configurable desde el registro de configuración del canal. Como se describe en la Tabla 2, se pueden seleccionar ganancias de 1 a 12. En la Fig. 14, se puede observar el circuito de implementación del PGA.

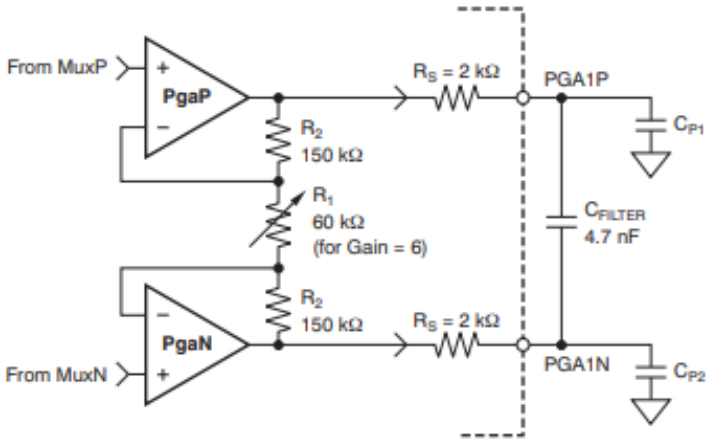


Fig. 14 Circuito de implementación del PGA

Antes de pasar al convertidor analógico digital, la salida del PGA es filtrada mediante un filtro de tipo RC (componentes R_S y C_{FILTER}) que funciona como un filtro anti-aliasing.

Al igual que la ganancia, se puede configurar mediante registros la frecuencia del reloj o la frecuencia de conversión de datos. En este caso, la frecuencia del reloj está configurada por defecto a 512 kHz y la frecuencia de conversión a 125 Hz, es decir 1 muestra cada 8 ms.

De este mismo modo, la ganancia tiene un valor por defecto de 6 para ambos canales.

La configuración de estos parámetros se realiza mediante la escritura de determinados registros del chip. En este proyecto se ha modificado sus valores a partir del código de Arduino (explicado en el apartado 3.4.1).

Los registros configurables son los siguientes:

- 0x00: Indica características del dispositivo.
- 0x01: Configura la frecuencia de muestreo del convertidor ADC y el tipo de conversión.
- 0x02: Configura la señal de referencia, la frecuencia de reloj y la señal de testeo.
- 0x03: Configura la operativa del Lead-Off (no se utiliza en este proyecto).
- 0x04: Configura el canal 1 del convertidor ADC (la ganancia y el tipo de electrodos).
- 0x05: Configura el canal 2 del convertidor ADC (la ganancia y el tipo de electrodos).
- 0x06: Configura el circuito RLD.
- 0x07: Configura los parámetros del Lead-Off.
- 0x08: Lee los parámetros determinados en el registro 0x07 y configura el divisor del reloj.
- 0x09: Configura la funcionalidad de la respiración.
- 0x0A: Configura la calibración de la funcionalidad de la respiración.

Una vez realizada la configuración de los parámetros del chip, está listo para captar los impulsos eléctricos y transmitirlos.

En la Fig. 15, se puede observar el protocolo de transmisión de los datos. Como la conversión de datos está configurada en modo continuo, indica cuando hay nuevos datos mediante el puerto DRDY (Data Ready).

Cuando el valor de este pin cambia de 1 a 0 hay datos nuevos. En el próximo flanco ascendente del SCLK (pulso del reloj) se transmitirán los datos por el puerto DOUT empezando por el MSB[4].

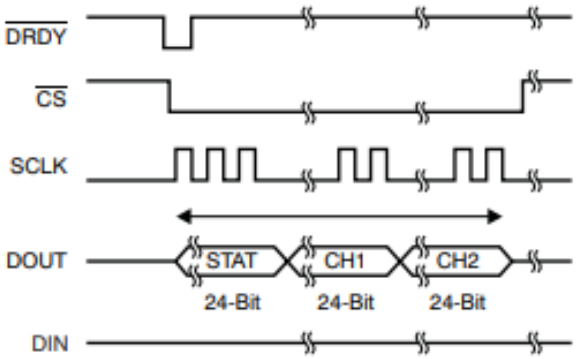


Fig. 15 Protocolo de transmisión de datos del chip ADS1292R

3.3.1.2. Transmisión de los datos

La trama que se envía por el pin DOUT está compuesta por tres paquetes de 24 bits cada uno.

Como se puede observar en la Tabla 3, los tres primeros bytes[5] corresponden a la cabecera, los tres siguientes corresponden al valor de la respiración (primer canal) y los tres últimos corresponden al valor del ECG (segundo canal).

MSB								LSB
0	1	2	3	4	5	6	7	8
24 bits			24 bits			24 bits		
CABECERA			CANAL 1			CANAL 2		

Tabla 3. Secuencia de bytes en la salida del convertidor ADS1292R

En total, en un envío se transmiten 72 bits. No hay bit de stop ni de paridad.

[4] Most Significant Bit
[5] 1 byte equivale a 8 bits.

- La cabecera:

Los bits de la cabecera tienen un valor constante que depende de bits almacenados en otras direcciones del chip. El valor está basado en la fórmula siguiente:

1100 (constante) + LOFF_STAT [4:0] + GPIO [1:0] + 13 '0's. (constante)

La dirección LOFF_STAT se encuentra en el registro 0x08 del chip y contiene la información relacionada con el estado de la conexión de los electrodos (positivos y negativos) en cada canal. La distribución de sus 8 bits se puede observar en la Tabla 4 siendo el bit 7 el más significativo.

BIT 7	**BIT 6**	**BIT 5**	**BIT 4**	**BIT 3**	**BIT 2**	**BIT 1**	**BIT 0**
0	CLK_DIV	0	RLD_STAT	IN2N_OFF	IN2P_OFF	IN1N_OFF	IN1P_OFF

Tabla 4. Extracto de la tabla 14 del datasheet del ADS1292R

Los bits que se utilizan como parte de la cabecera son del bit 4 al 0:

- El bit RLD_STAT (4) indica si el RLD (Right Leg Drive)[6] está conectado (0) o no (1).
- Los bits IN2N_OFF, IN2P_OFF, IN1N_OFF y IN1P_OFF nos indican si están conectados (0) o no (1), los electrodos positivos y negativos en el canal 2 y 1 respectivamente.

La otra dirección que interviene en la cabecera es la GPIO. Se encuentra en el registro 0x0B y contiene información sobre este pin.

Es un puerto que se puede usar como entrada o salida digital dependiendo del valor de los bits GPIOC2 (para el canal 2) y GPIOC1 (para el canal 1). Podemos observar la distribución de los 8 bits de esta dirección en la Tabla 5, siendo el bit 7 el más significativo.

BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
0	0	0	0	GPIOC2	GPIOC1	GPIOD2	GPIOD1

Tabla 5. Extracto de la tabla 14 del datasheet del ADS1292R

[6] Consiste en un circuito eléctrico diseñado para contrarrestar las interferencias en modo común de los amplificadores.

Teniendo en cuenta los valores de los bits de estas dos direcciones, la cabecera adquiere el valor en binario mostrado en la Tabla 6.

Valor constante	Bits de la dirección LOFF_STAT	Bits de la dirección GPIO	Valor constante (13 '0's)
1100	0000 0	00	0 0000 0000 0000

Tabla 6. Valor final de los bits de la cabecera.

- Bits del canal 1 y 2:

A continuación, encontramos los 24 bits del canal 1 y los 24 bits del canal 2.

Ambos canales transmiten los datos en formato **binario complemento a dos**. El rango de estos datos está comprendido entre 0x800000 a 0x7FFFFF, siendo el primer bit en recibirse el más significativo.

Como resume la Tabla 7, para una señal superior a VREF [7], es decir el valor máximo de entrada, en la salida obtenemos 0x7FFFFF, mientras que para una señal inferior a -VREF, en la salida obtendremos 0x800000.

INPUT SIGNAL, VIN (AINP – AINN)	OUTPUT CODE
≥ VREF	0x7FFFFF
≥ VREF / (2^{23} – 1)	0x000001
0	0x000000
–VREF / (2^{23} – 1)	0xFFFFFF
≤ –VREF (2^{23} / 2^{23} – 1)	0x800000

Tabla 7. Resumen de la relación entre el la señal de entrada y el código de salida.

[7] Tensión de referencia del convertidor ADS1292R

En la Fig. 16 se puede observar la escala de valores que puede tomar el código a la salida del convertidor:

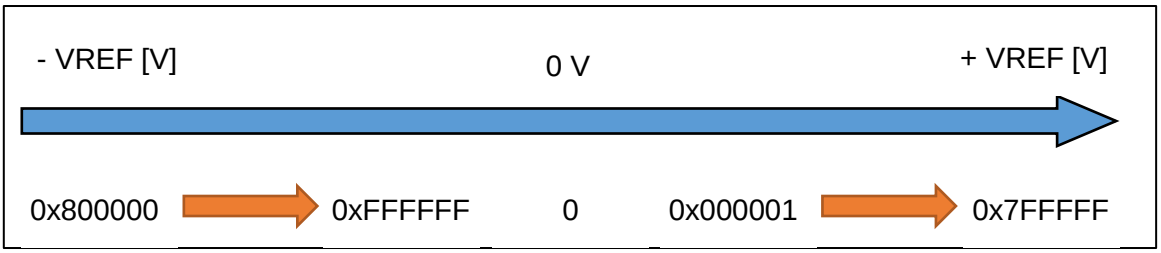


Fig. 16 Escala de valores del código de salida del convertidor y la señal de entrada.

3.3.2. Arduino

En la Fig. 17 se puede observar la placa microcontroladora Arduino UNO. Dispone de 13 pines digitales y 6 analógicos para usar como entradas o salidas.

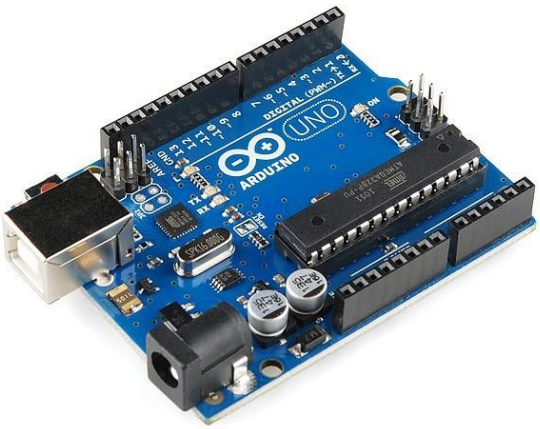


Fig. 17 Placa microcontroladora Arduino UNO

Se conecta directamente al ordenador mediante cable USB A/B para alimentarse y programarse. Además, cuenta con dos pines de alimentación a 5V o 3,3V.

Dispone de un software propio para su programación, así como el acceso libre a una infinita cantidad de librerías y códigos online.

Este es uno de los principales motivos por el cual se ha seleccionado esta placa microcontroladora para el proyecto. El otro motivo es su fácil conexión con el shield ADS1292R que se comentará en el apartado 3.4.1.

3.3.3. LabVIEW

LabVIEW es un software muy utilizado en el desarrollo de sistemas de medida, interfaces de usuario o SCADAS[8] tanto en la industria como para pequeñas aplicaciones.

Se ha seleccionado LabVIEW para el proyecto porque ofrece las siguientes ventajas respecto otros softwares:

1. La programación es sencilla ya que se realiza mediante bloques.
2. Incluye la visualización del programa y el código en dos ventanas distintas.
3. La implementación del filtrado y procesado de los datos requerido en este proyecto es sencilla.
4. La personalización del programa es muy extensa.

El software se basa en dos ventanas principales:

- Front Panel
- Block Diagram

La programación se lleva a cabo en el Block Diagram mientras que la visualización simultánea del programa ocurre en el Front Panel.

Como se puede observar en la Fig. 18, des de la paleta de funciones del Block Diagram se puede seleccionar los elementos que formarán parte del código.

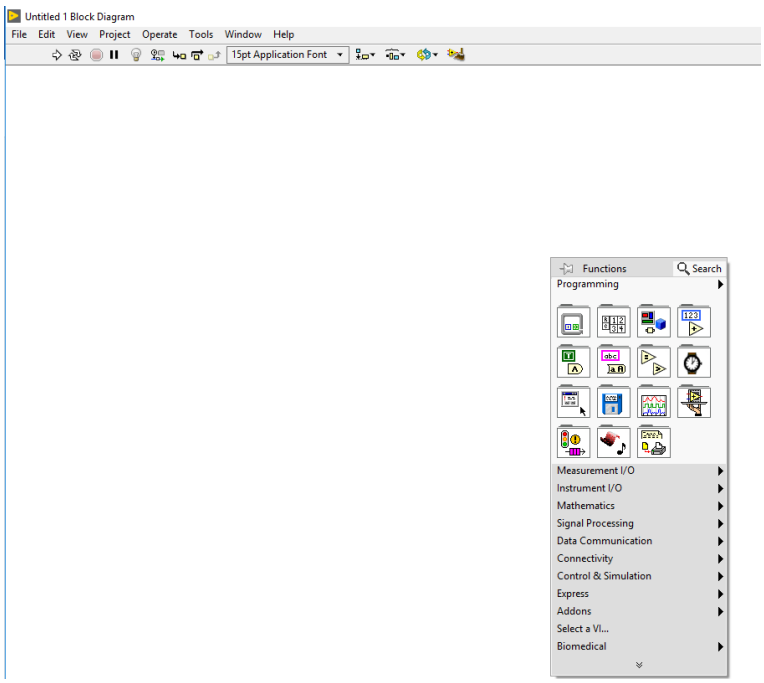


Fig. 18 La paleta de funciones de LabVIEW

[8] Supervisory Control And Data Acquisition

Si, por ejemplo, el elemento añadido es un botón, aparecerá automáticamente en el Front Panel, como se puede apreciar en la Fig. 19.

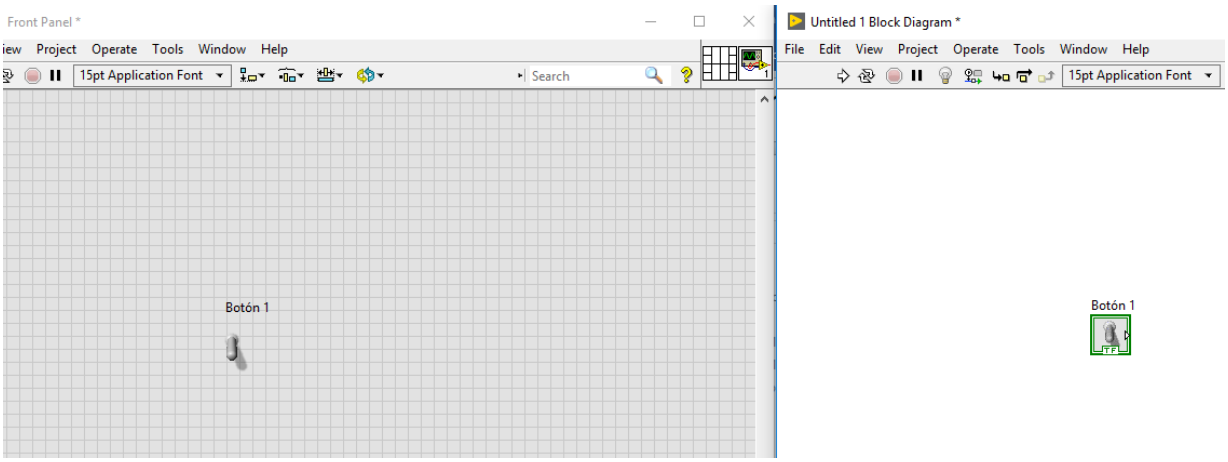


Fig. 19 El front panel y el block diagram

Para que los elementos interactúen entre sí, se deben unir mediante cables en el Block Diagram. En la Fig. 20, el botón anterior activa y desactiva un LED que se visualiza en el Front Panel.

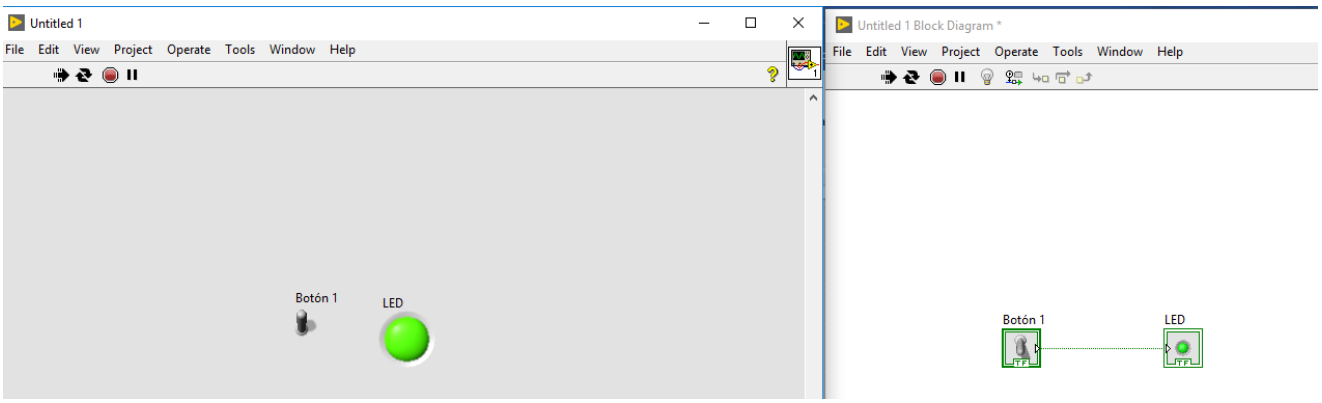


Fig. 20 Conexión entre los bloques del bloc diagram

De este mismo modo se pueden introducir estructuras de control como por ejemplo bucles While, For o de tipo Case que tienen un tratamiento similar a su programación en otros lenguajes.

Ofrece también la posibilidad de recibir y enviar datos mediante el puerto serie gracias a un controlador gratuito llamado VISA. Cuenta con varias funciones de control de flujo y tratamiento de la trama.

Por último, hay que destacar que LabVIEW ofrece herramientas para el filtrado de la señal en su software base. Sin embargo, existe un toolkit gratuito específicamente diseñado para la monitorización del ECG y su postratamiento. Las funciones más destacadas que ofrece son las siguientes:

- Filtrado y post procesado de la señal de ECG.
- Caracterización de la señal y detección del QRS
- Información acerca de la frecuencia cardíaca y las ondas en cada ciclo.

Para la total implementación de este toolkit es necesario hacer uso de otro que, en este caso no es gratuito. Se trata del Advanced Signal Processing Toolkit. Gracias a la licencia de la escuela se obtuvo una prueba de 60 días.
En este proyecto el filtrado se implementó mediante los bloques del software base, pero se hizo uso del bloque detector de ritmo cardíaco incluido en el toolkit.

3.4. Métodos

3.4.1. Shield ADS1292R y Arduino

La Fig. 21 muestra la conexión entre ambos elementos, como es habitual, se inserta en los pines de la placa Arduino.

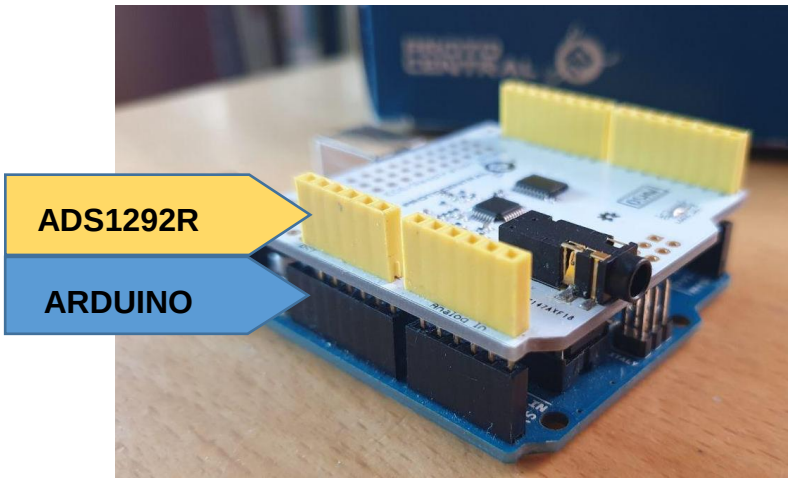


Fig. 21. Conexión entre el shield y la placa Arduino

La conexión de los pines entre el ADS1292R y Arduino se detalla en la Tabla 8.

Pin ADS1292R	Pin Arduino
CLK	Pin digital 3
PWDN/RESET	Pin digital 4
START	Pin digital 5
DRDY	Pin digital 6
CS	Pin digital 7
DIN	Pin digital 11
DOUT	Pin digital 12
SCLK	Pin digital 13

Tabla 8 Relación entre los pines del ADS1292R y el Arduino

La interacción entre ambos elementos se construye mediante la programación de Arduino desde su software. El código de está dividido en:

- Un sketch principal donde se encuentra el programa que se ejecuta en la placa.
- Un archivo de librería con extensión "*.h*" que contiene la declaración de los registros del chip y de los métodos como, por ejemplo, la inicialización o la activación del modo de lectura continuo.
- Un archivo de librería con extensión "*.cpp*" que contiene la escritura de los registros y la ejecución de los métodos declarados.

Estas dos librerías son de acceso libre y están publicadas por los usuarios de Protocentral. Se pueden obtener desde el repositorio de GitHub dedicado a este shield además de un programa de muestra desarrollado en lenguaje *processing*.

Tanto el sketch principal como las librerías se encuentran adjuntos para su consulta en el apartado Apéndices.

El sketch de Arduino donde se desarrolla el código principal se compone de dos partes.

- La sección *setup:*

Se ejecuta una sola vez al inicio del programa. En este caso contiene la función de inicialización del chip, donde se escriben los registros del chip mediante la instrucción:

"*ads_Reg_Write(ADS1292_REG_**Nombre_del_registro**¸VALOR);"*

De este modo escribimos los 11 registros de configuración del chip.

- La sección *loop*:

Se lleva a cabo una secuencia de operaciones descritas en el diagrama de la Fig. 22.

Debido a que el chip está configurado en modo de lectura continuo, nos indica cuando hay nuevas muestras por leer a partir del pin DRDY, cuando su valor cambia de 1 a 0.

A continuación, se leen los datos del registro y se mueven a una variable de tipo *unsigned long*[9], ya en el sketch principal de Arduino. Como los valores de tensión del ECG pueden ser positivos o negativos, se mueve este número a una variable del tipo *signed long*[10].

El mismo procedimiento para el canal 2.

[9] Tipo de variable numérica con un rango de valores comprendido entre 0 y 4.294.967.295
[10] Tipo de variable numérica con un rango de valores comprendido entre -2.147.483.648 y 2.147.483.647

Tras esta conversión se obtiene un array de dos posiciones con los 3 bytes de cada canal.

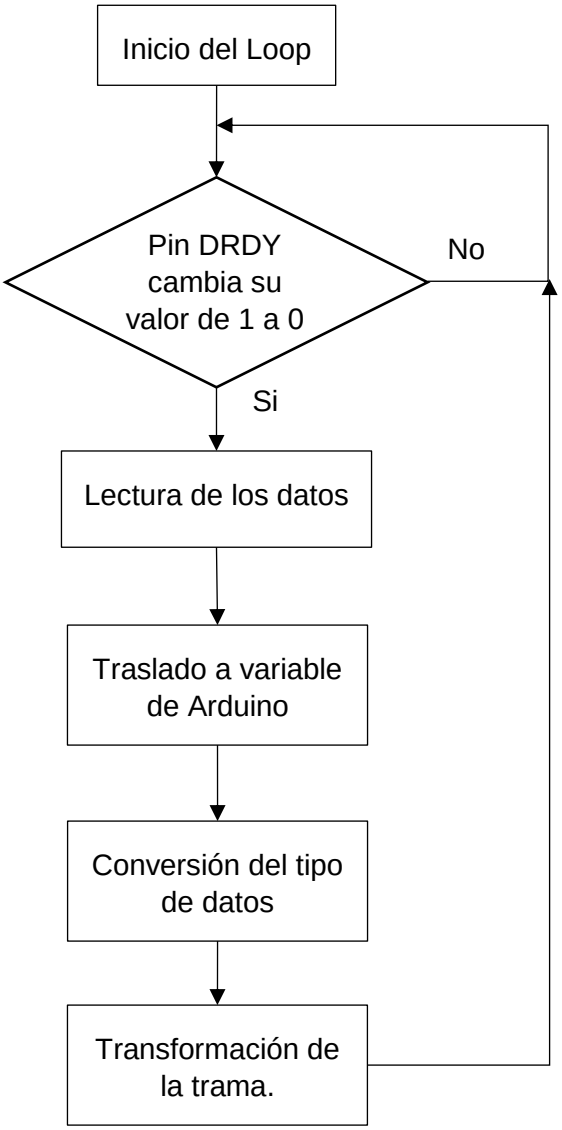


Fig. 22 Diagrama de flujo de la sección void loop

Finalmente, antes de enviar los datos por el puerto serie, se construye una nueva trama con 2 bytes de detección de inicio y 1 byte de detección de final como muestra la Tabla 9.

B0	Bit de Inicio 1 definido como 0x0A	***LSB de la trama***
B1	Bit de Inicio 2 definido como 0xFA	
B2	Bit 1 del ECG (MSB)	
B3	Bit 2 del ECG	
B4	Bit 3 del ECG (LSB)	
B5	Bit 1 de la respiración (MSB)	
B6	Bit 2 de la respiración	
B7	Bit 3 de la respiración (LSB)	
B8	Bit de Stop definido en 0x0B	***MSB de la trama***

Tabla 9 Estructura de la trama construida en Arduino

La cabecera inicial proveniente del chip ADS1292R se pierde en esta transformación de los datos.

3.4.2. LabVIEW

La interfaz desarrollada mediante el software LabVIEW se divide en dos pantallas principales:

- Configuración: Permite modificar el puerto serie del Arduino, la velocidad de transmisión o la ruta para guardar los datos como se observa en la Fig. 23.
- Análisis: Visualización del ECG filtrado o no a elección del usuario, las pulsaciones por minuto y la respiración de este como se observa en la Fig. 24.

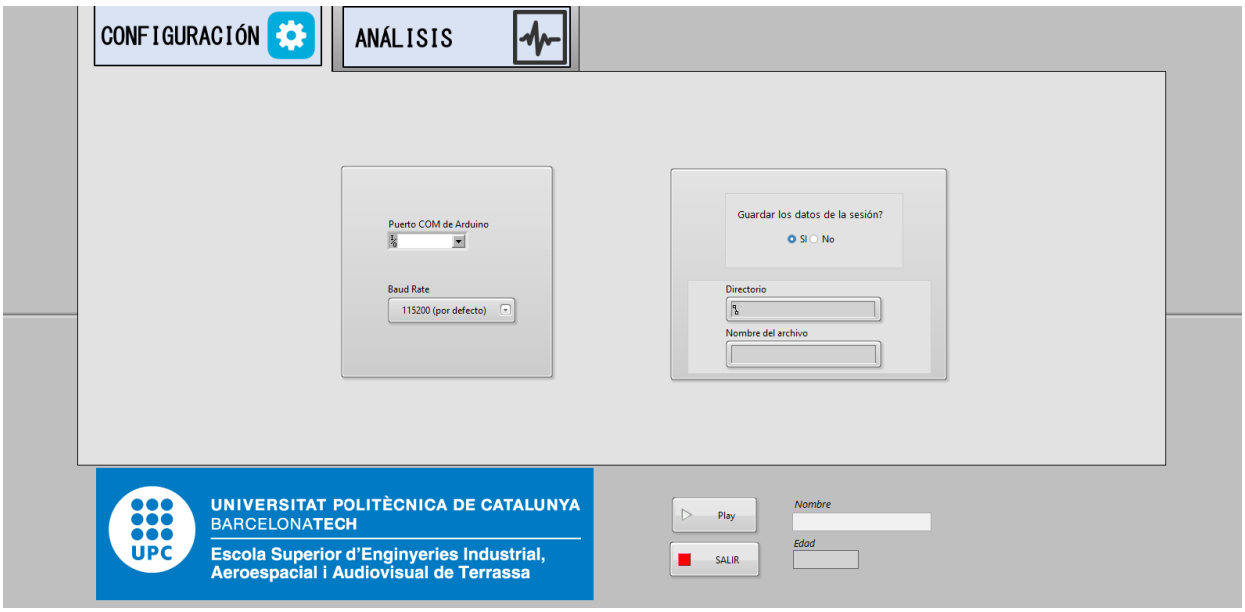


Fig. 23 Pantalla de configuración de la interfaz

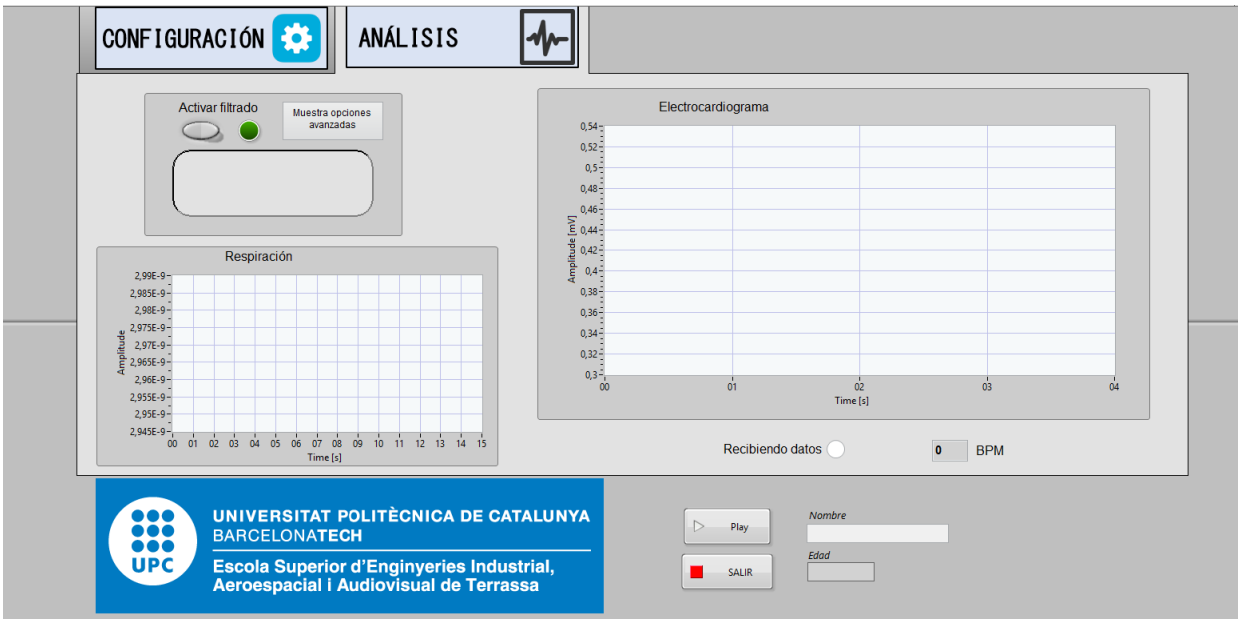


Fig. 24 Pantalla del análisis de la interfaz

Antes de iniciar el programa se debe seleccionar en la pantalla de configuración, el puerto COM del Arduino y si se quiere guardar los datos de la sesión o no. De lo contrario el programa devuelve un error.

Para iniciar o detener el programa se deben pulsar los botones de Play y Salir respectivamente.

Una vez iniciado, el programa se preguntará el nombre y la edad del usuario para añadirlas a la ventana. En una futura implementación se usarán estos datos para crear un registro clasificado por edades y posiblemente por sexo. El código de LabVIEW se encuentra adjunto también en el apartado 7.1.

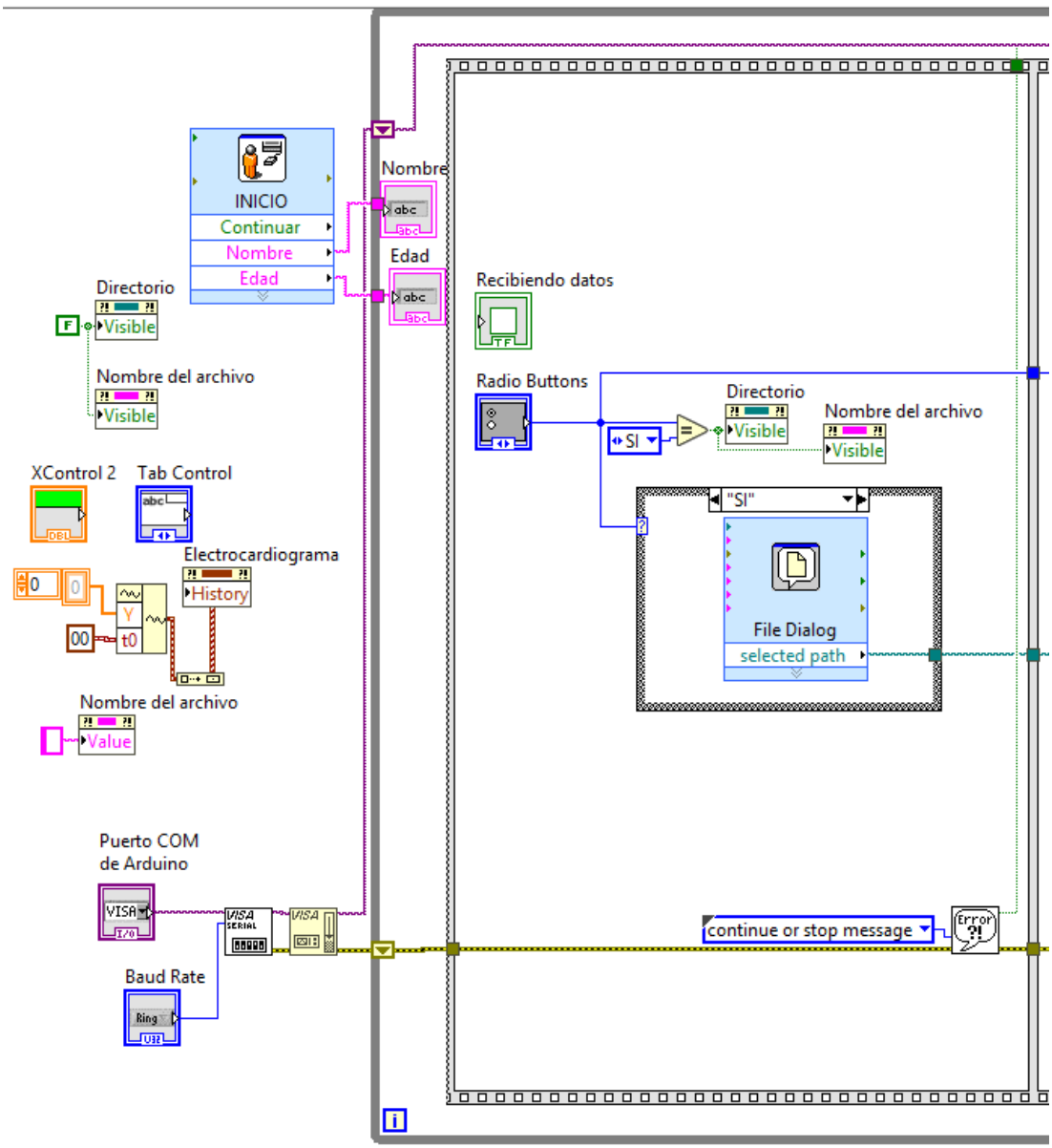


Fig. 25 Código del block diagram (I)

Como se observa en la Fig. 25, en la parte de la izquierda se configuran y reinician las variables del programa. Si nos desplazamos hacia la derecha, el código continúa y comprueba si se ha seleccionado la opción de guardar los datos, en ese caso pregunta la ubicación del archivo. Al mismo tiempo si existe un error en la configuración, aparecerá un cuadro de diálogo con información del error y dando la posibilidad de detener el programa.

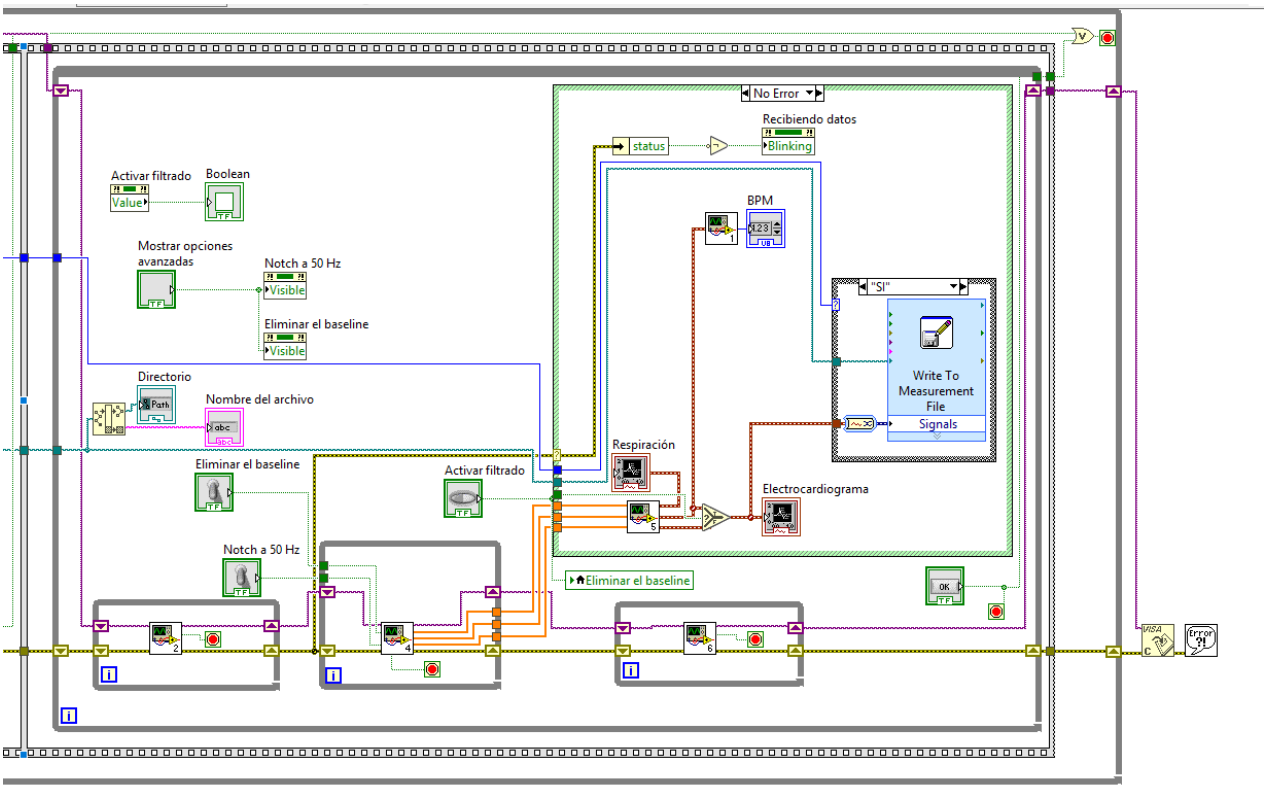


Fig. 26 Código del block diagram (II)

En la Fig. 26 se observa la continuación del programa. En esta parte se encuentra la adquisición y tratamiento de los datos, además de otras pequeñas configuraciones de la pantalla de Análisis del Front Panel.

En la parte superior izquierda se encuentran las activaciones de los indicadores luminosos o la visibilidad de los botones de filtrado.

En la parte superior derecha se encuentran los gráficos para el ECG y para la respiración. También se encuentra la SubVI del cálculo de las pulsaciones por minuto.

Para guardar los datos hay una estructura de tipo case que lee la configuración del selector de la pantalla de configuración. En el caso que se deban guardar los datos, se ha usado el bloque *Write To Measurement File* para generar el archivo en la ruta especificada por el usuario. Por el momento solo recoge los datos del ECG.

El código que trata y filtra los datos recibidos está comprimido en SubVIs. Son pequeños subprogramas que funcionan como un bloque en sí y disponen de entradas y salidas de variables. A continuación, se describirán las 5 SubVIs que incluye el programa:

1. SubVI – Detección de inicio de la trama:

En la Fig. 27 se observa el código de la detección de inicio. Se comprueba si hay bits en el puerto serie, se leen 2 veces 1 byte y se almacena en un String concatenado. A continuación, se compara con el String de referencia (0AFA). Si coincide se detiene el bucle y se continua el programa.

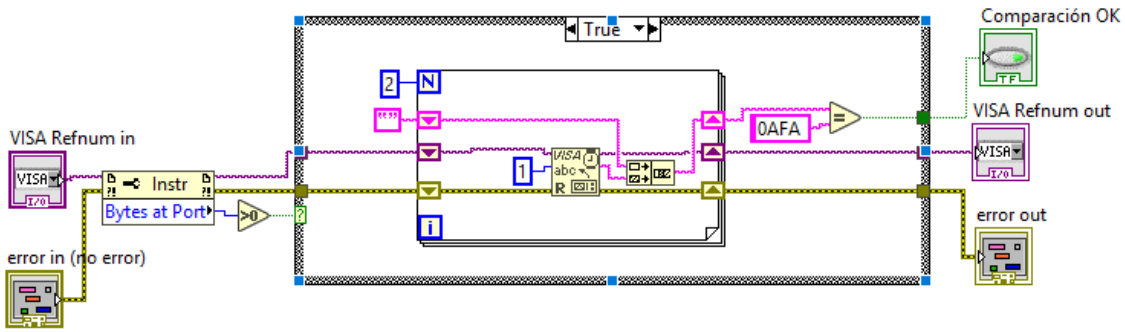


Fig. 27 Código de la detección de inicio incluido en la SubVI1

2. SubVI – Tratamiento de los datos y filtrado

En la Fig. 28 y *Fig. 29* se observa el código correspondiente al tratamiento y filtrado de los datos recibidos para ambos canales.

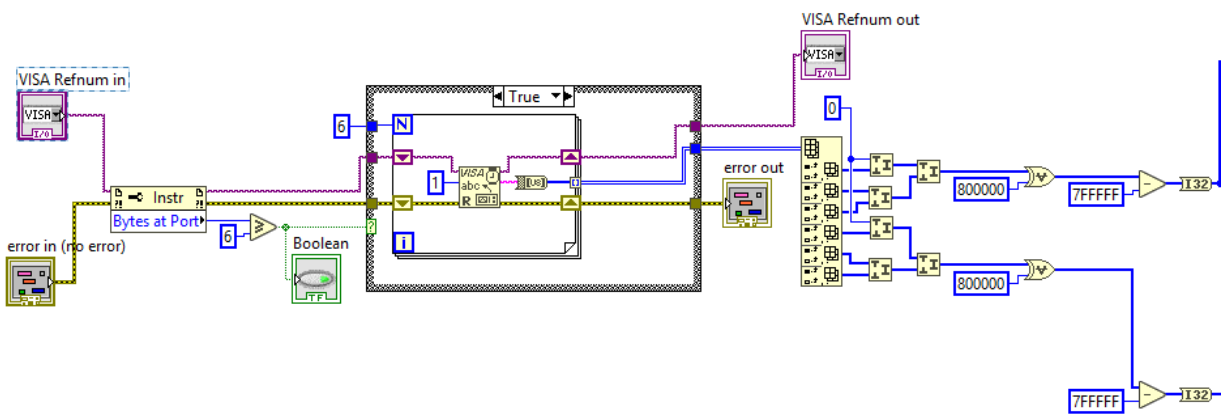


Fig. 28 Código (I) del tratamiento y filtrado de los datos en la SubVI 2

Igual que en la SubVI anterior, se comprueba que haya bits en el puerto serie, en este caso se leen 6 veces 1 byte y se almacenan en un array de bytes.

A continuación, se reúnen los bytes en cada canal obteniendo un único valor de 32 bytes.

Para modificar el rango de este valor de 0x800000 – 0x7FFFFF a 0x000000 – 0xFFFFFF se utiliza una puerta lógica XOR entre el valor y 0x800000.

Finalmente se resta el valor de V_{REF} para centrar el dato, en este caso sería 2^{23} -1 = 0x7FFFFF.

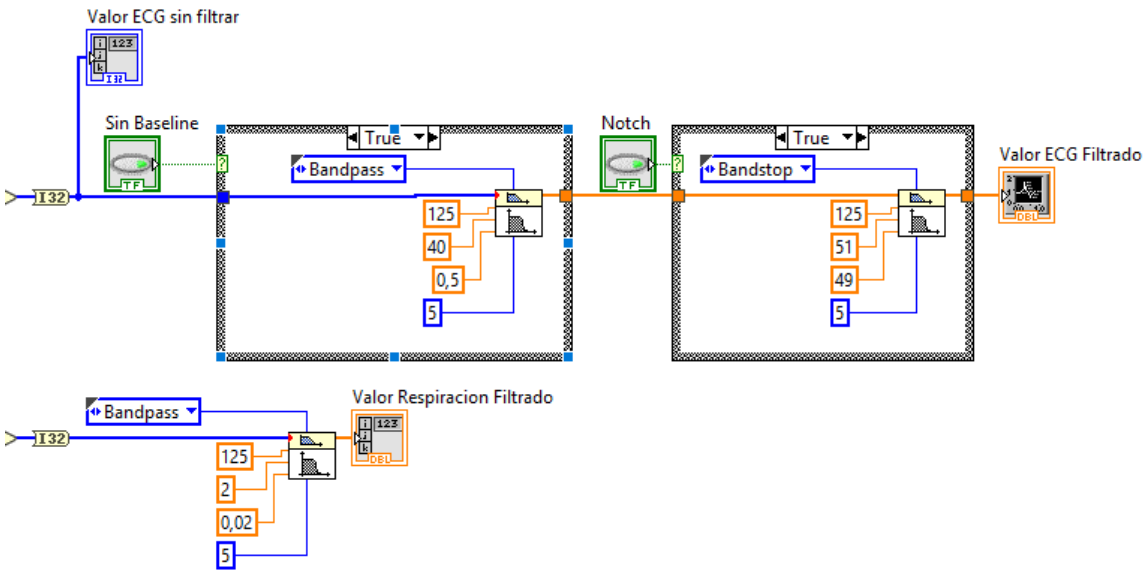


Fig. 29 Código (II) del tratamiento y filtrado de los datos en la SubVI 2

Finalmente, en la segunda parte del código se realiza el filtrado de la señal. Como ya se ha comentado, el ECG tiene un rango de frecuencias de 0,5 Hz a 100 Hz, pero solamente se filtrará hasta los 40 Hz. Se ha implementado mediante un filtro pasa banda de orden 5.

A continuación, se aplica un filtro rechaza banda de orden 5 situado en los 50 Hz para atenuar las señales de la red eléctrica.

En el caso de la respiración se aplica un filtro pasa banda de orden 5 en un rango de frecuencias de 0,02 Hz a 2 Hz.

3. SubVI – Conversión de los valores:

En esta SubVI se convertirán los valores digitales a valores de tensión. Como se observa en la Fig. 30, para realizar esta conversión se aplica la siguiente fórmula:

Tensión [mV] = Código (decimal) x (V_{REF} x 1000) / (Ganancia x (2^{23} -1))

En el caso de la respiración el procedimiento es el mismo ya que la única posible variación sería la ganancia, pero ambos canales están configurados al mismo valor.

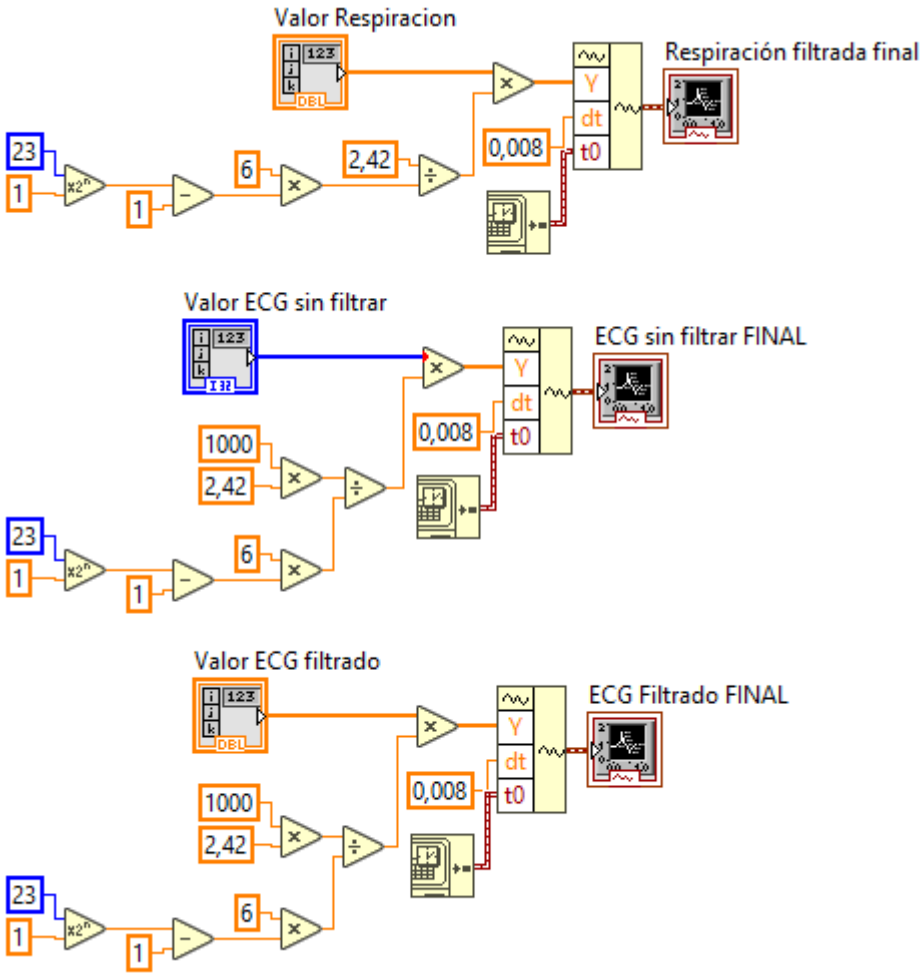


Fig. 30 Conversión de los valores digitales a tensión en la SUBVI

4. SubVI – Cálculo de las pulsaciones por minuto

El código de esta SubVI está detallado en la *Fig. 31*. Para poder determinar las pulsaciones por minuto se debe establecer un valor límite (threshold) con el que se pueda diferenciar el pico del complejo QRS del resto de ondas.

Esta operación se realiza a partir del valor máximo del ECG determinado mediante una estructura de tipo *formula node* donde realizamos la comparación entre la señal y el valor máximo y la consiguiente actualización de este.

Una vez determinado el valor máximo de la señal, se aplican unos factores para obtener el límite superior y el inferior entre los cuales se encuentra el pico del complejo QRS. Estos factores se han obtenido de forma empírica.

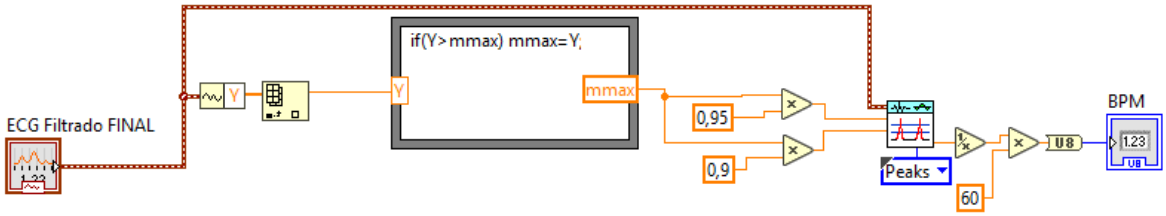


Fig. 31 SubVI para el cálculo de las pulsaciones por minuto

Para determinar las pulsaciones por minuto se ha usado el bloque Biosignal Rate Extractor, incluido en el Biomedical Toolkit. A partir de estos dos valores límite, devuelve el tiempo entre cada pico de complejos QRS. Para obtener los picos por segundo dividimos 1 entre este valor y multiplicamos por 60 para obtener los picos por minuto.

5. SubVI – Detección de final

Por último, para detectar el final de la trama se ha implementado un código detallado en la Fig. 32. Comprueba si hay bits en el puerto, lee 1 byte y lo compara con el byte de referencia (0x0B). En caso de coincidir, detiene el bucle y continua el programa principal.

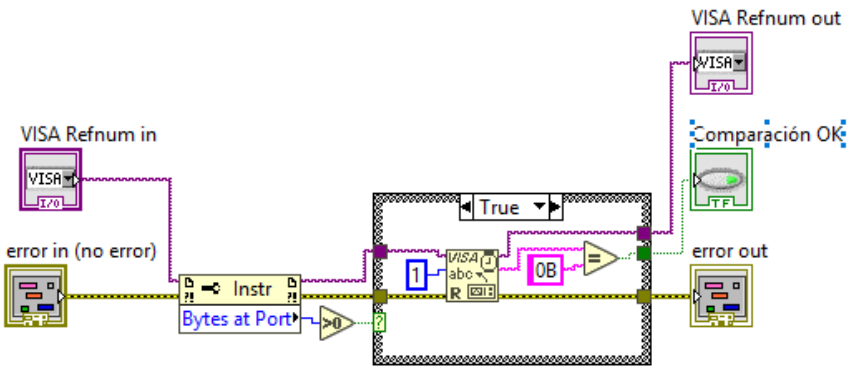


Fig. 32 Detección de final en la SubVI

4. Resultados

4.1. Prueba con los electrodos desechables incluidos en el shield

Para la primera prueba se usaron los electrodos desechables que venían incluídos con el shield. Los resultados del electrocardiograma y la respiracion fueron los siguientes:

- Electrocardiograma:

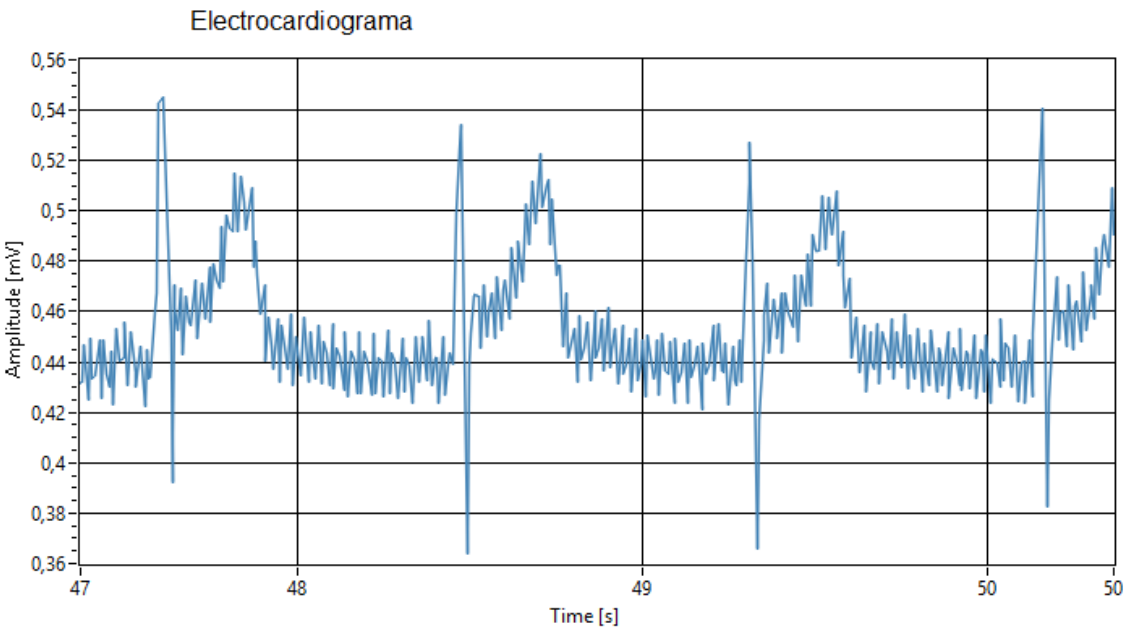


Fig. 33 Representación del ECG mediante los electrodos desechables incluidos con el shield

Como se puede observar en la Fig. 33, estos electrodos no son de gran calidad y la señal recogida contiene bastante ruido para ser analizada correctamente.

Sin embargo, al realizar la prueba no estaban siendo usados por primera vez.

- Respiración:

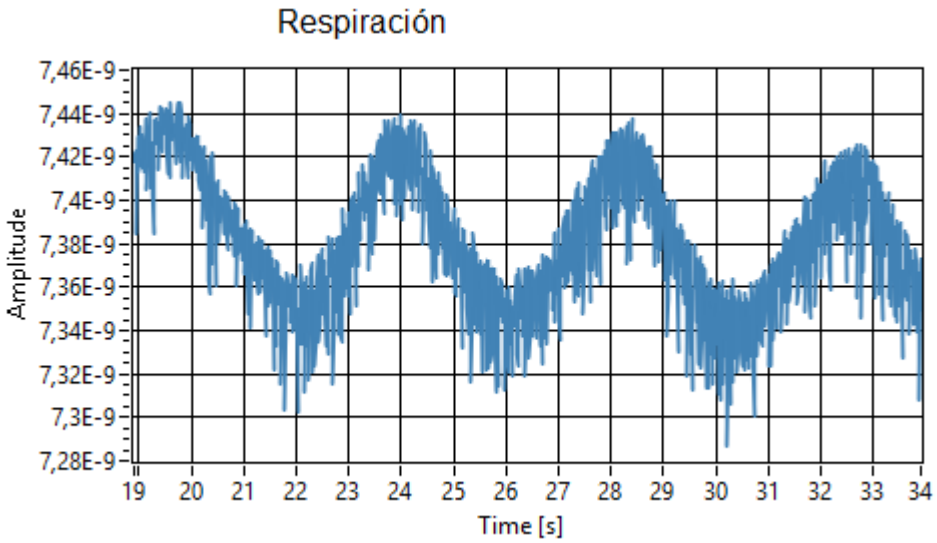


Fig. 34 Respiración mediante los electrodos desechables incluidos con el shield

En la Fig. 34 se representan 3 ciclos completos de respiración. La señal obtenida contiene aún algo de ruido. La magnitud de la amplitud de la respiración son Ω.

4.2. Prueba con los electrodos desechables comprados

A continuación se hizo la prueba con unos electrodos también en parches desechables pero de mayor calidad. Se pueden observar en la Fig. 35.

Fig. 35 Electrodos desechables comprados

Los resultados fueron los siguientes:

- Electrocardiograma.

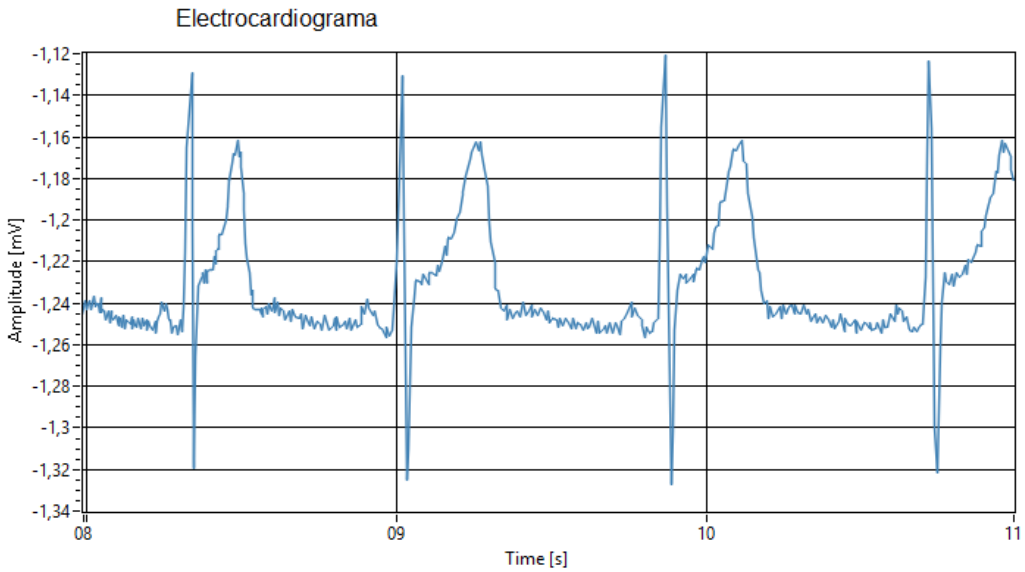


Fig. 36 ECG mediante los electrodos desechables

En la Fig. 36 se observa una representación nítida de las ondas principales del ECG. Sin embargo no se ha conseguido eliminar el pico de tensión negativo de la onda S.

- Respiración:

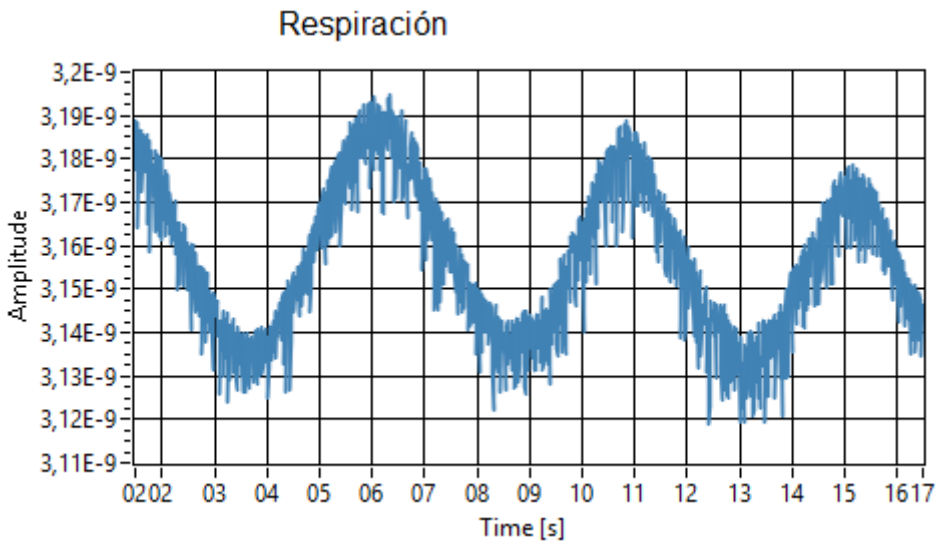


Fig. 37 La respiración mediante los electrodos desechables.

En la Fig. 37 se observan 3 ciclos completos de respiración. La magintud de la respiración está expresada en Ω. La señal contiene bastante ruido, a pesar de haber aplicado el filtrado.

4.3. Prueba con dos bandas elásticas y bordado conductor

Finalmente se hizo la prueba con unas bandas elásticas que incorporan un bordado con hilo conductor. En este segmento se ha insertado un broche para poder conectar los electrodos. En la Fig. 38 observamos las bandas que han sido prestadas por la escuela.

Fig. 38 Bandas con bordado hecho de hilo conductor

Se obtiene las siguientes representaciones:

- Electrocardiograma

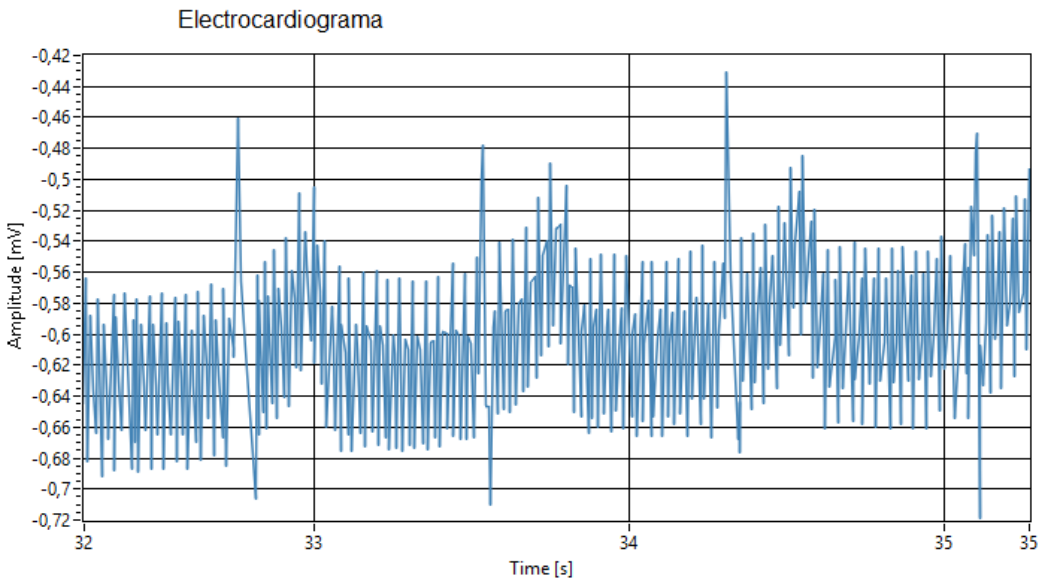


Fig. 39 Representación del ECG con las bandas elásticas

En la Fig. 39 se aprecian los elementos representativos del ECG, pero sin lugar a dudas la conductividad de los electrodos bordados no es la más elevada de estos tres casos. Se limpió la piel y la zona bordada con agua para aumentar esta conductividad.

- Respiración:

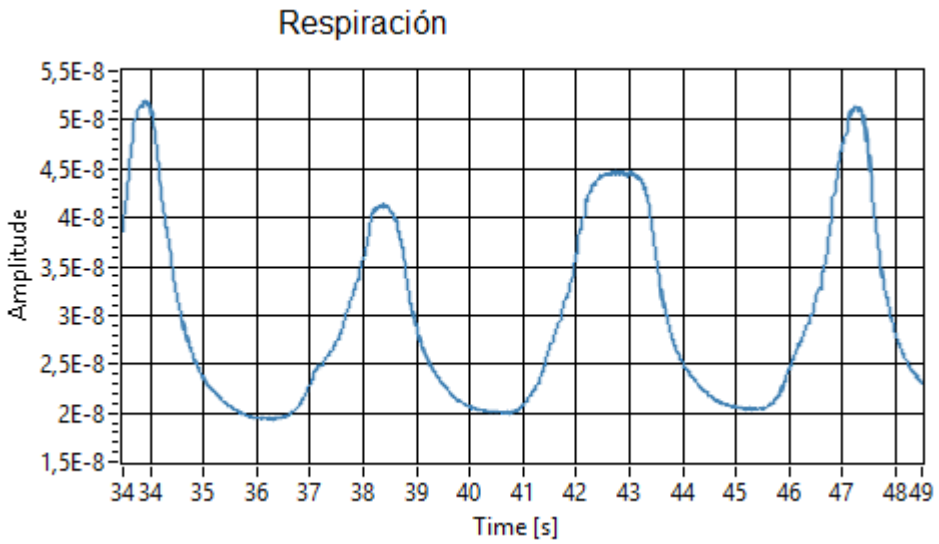


Fig. 40 Respiración mediante las bandas elásticas

En la Fig. 40 se observan 3 ciclos completos de respiración. A pesar de tener la peor representación del ECG de los tres casos, es la mejor en la respiración.

5. Conclusiones

Al inicio del proyecto se propuso como principal objetivo la correcta visualización del electrocardiograma. Como se ha podido observar en la Fig. 36 la representación obtenida para los electrodos desechables es bastante correcta. En un escenario superior se debería haber obtenido una señal perfectamente centrada en los 0V para los segmentos isoeléctricos y una onda S sin apenas pico de tensión. No obstante, los resultados para el ECG son satisfactorios.

Respecto a la respiración, es un añadido al proyecto ya que no forma parte del objetivo principal, sin embargo, su representación es correcta, aunque no lo sea su claridad.

La interfaz diseñada mediante LabVIEW proporciona una visualización clara de ambas señales. Permite configurar el dispositivo y exportar los datos a un archivo externo. Además, el aprendizaje sobre la utilización del puerto serie y el filtrado ha sido muy satisfactorio.

Los resultados del proyecto por lo tanto son satisfactorios respecto a los objetivos a posteriori.

5.1. Trabajos futuros

Las implementaciones más inmediatas serían sustituir la comunicación mediante Arduino y PC por una de tipo inalámbrica, como por ejemplo Bluetooth. Esta opción es sencilla gracias a los módulos HC diseñados para usar con Arduino, que con solamente 4 pines son capaces de realizar la comunicación.

Finalmente, una implementación más ambiciosa sería trasladar la interfaz de LabVIEW a una aplicación para smartphones siguiendo las tendencias actuales en este tipo de dispositivos portátiles y mediante comunicación Bluetooth.

6. Referencias bibliográficas

[1] "¿Cómo funciona el corazón? | CardioSalud." [Online]. Available: http://www.cardiosalud.org/corazon-y-salud/como-funciona-el-corazon. [Accessed: 10-Feb-2019].

[2] A. L. Marín, "Diseño y Desarrollo de un Sistema Ambulatorio para la Adquisición de Señales Cardíaca y Respiratoria," Sevilla, 2015.

[3] A. Sgarlatta, "Sensor inalámbrico de ECG conectado vía Bluetooth a aplicación de análisis automático en el teléfono móvil," Córdoba, 2016.

[4] "Ondas del Electrocardiograma." [Online]. Available: http://www.my-ekg.com/generalidades-ekg/ondas-electrocardiograma.html. [Accessed: 12-May-2019].

[5] "Electrocardiograma." [Online]. Available: https://es.wikipedia.org/wiki/Electrocardiograma. [Accessed: 23-Mar-2019].

[6] "FAPap - El-electrocardiograma." [Online]. Available: https://fapap.es/articulo/134/el-electrocardiograma. [Accessed: 28-Apr-2019].

[7] "Electrocardiograma ECG/EKG - Fundación Española del Corazón." [Online]. Available: https://fundaciondelcorazon.com/informacion-para-pacientes/metodos-diagnosticos/electrocardiograma.html. [Accessed: 10-Feb-2019].

[8] Inerciauruguay, "(77) EL CORAZON QUE ES Y SU FUNCIONAMIENTO CICLO CARDIACO DIASTOLE SISTOLE ANIMACION BIEN EXPLICADO - YouTube," 2017.

[9] "NI Device Drivers 2014.08 Part 1 - Windows 8, Windows 7, Windows XP (SP3), Windows Server 2008 R2, Windows Server 2003 R2 - National Instruments." [Online]. Available: http://www.ni.com/download/ni-device-drivers-august-2014/4818/en/. [Accessed: 27-Jan-2019].

[10] Microcontrollers Lab, "Arduino with Labview: Getting Arduino data through serial

communication in labview," 2017. [Online]. Available: http://microcontrollerslab.com/arduino-labview-data-serial-communication/. [Accessed: 27-Jan-2019].

[11] "6 claves para aprender a interpretar el electrocardiograma." [Online]. Available: https://www.elsevier.com/es-es/connect/medicina/6-claves-para-aprender-a-interpretar-el-electrocardiograma. [Accessed: 28-Apr-2019].

[12] "ADS1x9xECG-FE Demonstration Kit User's Guide," 2011.

[13] "LabVIEW for ECG Signal Processing - National Instruments." [Online]. Available: http://www.ni.com/tutorial/6349/en/. [Accessed: 21-Apr-2019].

[14] A. Purohit, "Calculate ECG Parameters through Labview."

[15] "Electrocardiography: Overview, ECG Indications and Contraindications, Preparation." [Online]. Available: https://emedicine.medscape.com/article/1894014-overview#showall. [Accessed: 20-Apr-2019].

[16] "Electrocardiograma (ECG)." [Online]. Available: https://www.mayoclinic.org/es-es/tests-procedures/ekg/about/pac-20384983.

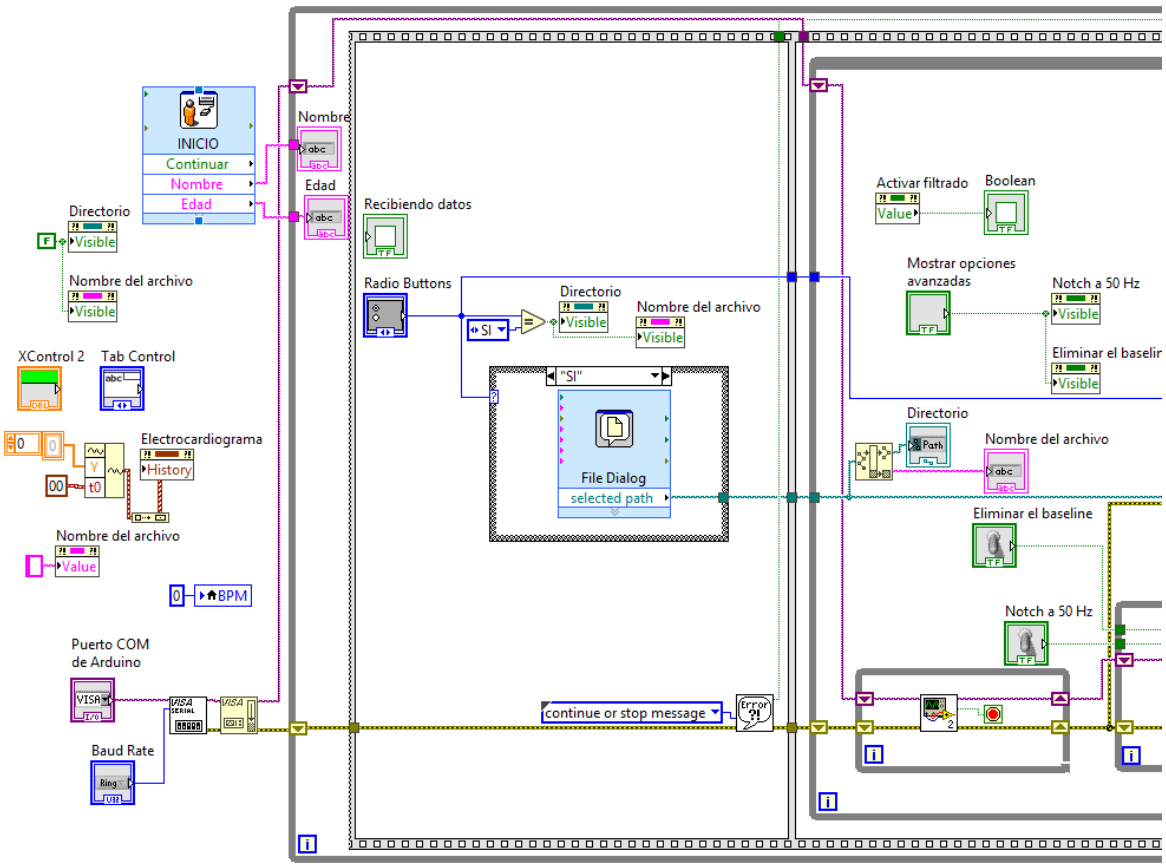
[17] J. M. Gayet, "Design of an Electrocardiography and Respiration Device for Ambulatory Monitoring."

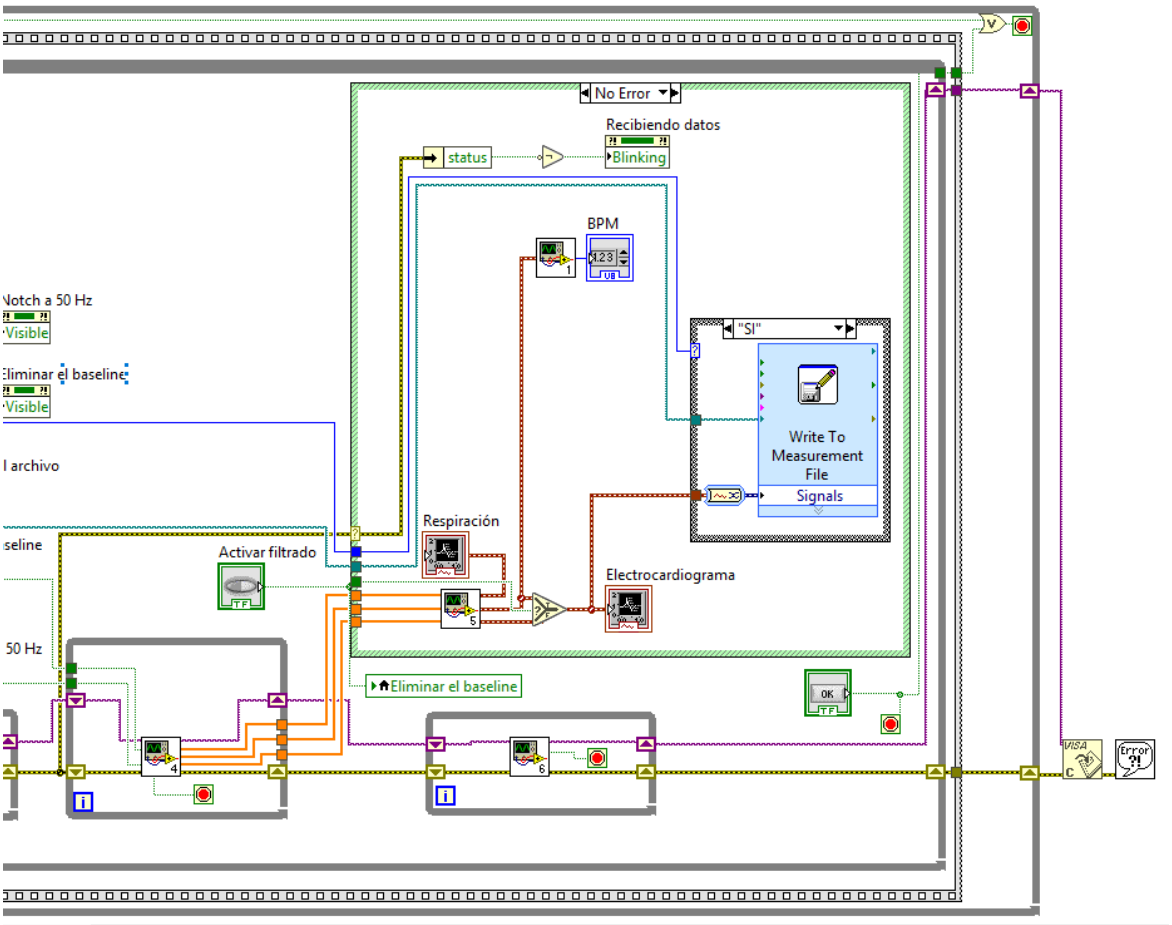
[18] "Experiment: Heart Action Potentials." [Online]. Available: https://backyardbrains.com/experiments/heartrate. [Accessed: 10-Feb-2019].

[1], [5]–[18]

7. Apéndices

7.1. Código de LabVIEW





7.2. Código de la placa Arduino

7.2.1. Sketch principal:

```
////////////////////////////////////////////////////////////////////////
//////////////////////
//
//    Arduino Library for ADS1292R Shield/Breakout
//
//    Copyright (c) 2017 ProtoCentral
//
//    This software is licensed under the MIT
License(http://opensource.org/licenses/MIT).
//
//    THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY
KIND, EXPRESS OR IMPLIED, INCLUDING BUT
//    NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR
A PARTICULAR PURPOSE AND NONINFRINGEMENT.
//    IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE
FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY,
//    WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
FROM, OUT OF OR IN CONNECTION WITH THE
//    SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
//
```

```
//   Requires g4p_control graphing library for processing.  Built
on V4.1
//   Downloaded from Processing IDE Sketch->Import Library->Add
Library->G4P Install
//
/////////////////////////////////////////////////////////////////
//////////////////////


#include <SPI.h>
#include <ads1292r.h>
ads1292r ADS1292;   // define class

//Packet format
#define  CES_CMDIF_PKT_START_1    0x0A
#define  CES_CMDIF_PKT_START_2     0xFA
#define  CES_CMDIF_TYPE_DATA       0x02
#define  CES_CMDIF_PKT_STOP        0x0B

volatile uint8_t  SPI_Dummy_Buff[30];
uint8_t DataPacketHeader[16];
volatile signed long s32DaqVals[8];
uint8_t data_len = 8;
volatile byte SPI_RX_Buff[15] ;
volatile static int SPI_RX_Buff_Count = 0;
volatile char *SPI_RX_Buff_Ptr;
volatile bool ads1292dataReceived =false;
unsigned long uecgtemp = 0;
signed long secgtemp=0;
int i,j;
long bitread;

void setup()
{
  // initalize the  data ready and chip select pins:
  pinMode(ADS1292_DRDY_PIN, INPUT);  //6
  pinMode(ADS1292_CS_PIN, OUTPUT);    //7
  pinMode(ADS1292_START_PIN, OUTPUT);  //5
  pinMode(ADS1292_PWDN_PIN, OUTPUT);  //4

  Serial.begin(115200);  // Baudrate for serial communica

  ADS1292.ads1292_Init();  //initalize ADS1292 slave
}

void loop()
{
  if((digitalRead(ADS1292_DRDY_PIN)) == LOW)       // Sampling
rate is set to 125SPS ,DRDY ticks for every 8ms
  {
    SPI_RX_Buff_Ptr = ADS1292.ads1292_Read_Data(); // Read the
data,point the data to a pointer

    for(i = 0; i < 9; i++)
```

```
    {
      SPI_RX_Buff[SPI_RX_Buff_Count++] = *(SPI_RX_Buff_Ptr + i);
// store the result data in array
    }
    ads1292dataReceived = true;

  }


  if(ads1292dataReceived == true)       // process the data
  {
    j=0;
    for(i=0;i<6;i+=3)                  // data outputs is (24
status bits + 24 bits Respiration data +  24 bits ECG data)
    {

        uecgtemp = (unsigned long) (  ((unsigned
long)SPI_RX_Buff[i+3] << 16) | ( (unsigned long) SPI_RX_Buff[i+4]
<< 8) |  (unsigned long) SPI_RX_Buff[i+5]);
        uecgtemp = (unsigned long) (uecgtemp << 8);
        secgtemp = (signed long) (uecgtemp);
        secgtemp = (signed long) (secgtemp >> 8);

        s32DaqVals[j++]=secgtemp;
    }

    DataPacketHeader[0] = CES_CMDIF_PKT_START_1;
    DataPacketHeader[1] = CES_CMDIF_PKT_START_2;

    DataPacketHeader[2] = s32DaqVals[1]>>16;            // 4 bytes
ECG data
    DataPacketHeader[3] = s32DaqVals[1]>>8;
    DataPacketHeader[4] = s32DaqVals[1];

    DataPacketHeader[5] = s32DaqVals[0]>>16;            // 4 bytes
Respiration data
    DataPacketHeader[6] = s32DaqVals[0]>>8;
    DataPacketHeader[7] = s32DaqVals[0];

    DataPacketHeader[8] = CES_CMDIF_PKT_STOP;

    for(i=0; i<9; i++)
    {
      Serial.write(DataPacketHeader[i]);     // transmit the data
over USB

     }
   }
    ads1292dataReceived = false;
    SPI_RX_Buff_Count = 0;
}
```

7.2.2. Cabecera de la librería:

```
////////////////////////////////////////////////////////////////////
////////////////////////
//
//   Arduino Library for ADS1292R Shield/Breakout
//
//   Copyright (c) 2017 ProtoCentral
//
//   This software is licensed under the MIT
License(http://opensource.org/licenses/MIT).
//
//   THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY
KIND, EXPRESS OR IMPLIED, INCLUDING BUT
//   NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR
A PARTICULAR PURPOSE AND NONINFRINGEMENT.
//   IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE
FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY,
//   WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
FROM, OUT OF OR IN CONNECTION WITH THE
//   SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
//
//   Requires g4p_control graphing library for processing.  Built
on V4.1
//   Downloaded from Processing IDE Sketch->Import Library->Add
Library->G4P Install
//
////////////////////////////////////////////////////////////////////
////////////////////////


#ifndef ads1292r_h
#define ads1292r_h

#include "Arduino.h"

#define CONFIG_SPI_MASTER_DUMMY   0xFF

// Register Read Commands
#define  RREG    0x20;          //Read n nnnn registers starting
at address r rrrr
                                //first byte 001r rrrr (2xh)(2) -
second byte 000n nnnn(2)
#define WREG    0x40;      //Write n nnnn registers starting at
address r rrrr
                                //first byte 010r rrrr (2xh)(2) -
second byte 000n nnnn(2)

#define START        0x08       //Start/restart (synchronize)
conversions
#define STOP         0x0A       //Stop conversion
#define RDATAC      0x10        //Enable Read Data Continuous
mode.
```

```
//This mode is the default mode at power-up.
#define SDATAC        0x11       //Stop Read Data Continuously
mode
#define RDATA         0x12       //Read data by command; supports
multiple read back.

//Pin declartion the other you need are controlled by the SPI
library
const int ADS1292_DRDY_PIN = 6;
const int ADS1292_CS_PIN = 7;
const int ADS1292_START_PIN = 5;
const int ADS1292_PWDN_PIN = 4;

//register address
#define ADS1292_REG_ID                0x00
#define ADS1292_REG_CONFIG1           0x01
#define ADS1292_REG_CONFIG2           0x02
#define ADS1292_REG_LOFF         0x03
#define ADS1292_REG_CH1SET       0x04
#define ADS1292_REG_CH2SET       0x05
#define ADS1292_REG_RLDSENS           0x06
#define ADS1292_REG_LOFFSENS     0x07
#define ADS1292_REG_LOFFSTAT     0x08
#define ADS1292_REG_RESP1       0x09
#define ADS1292_REG_RESP2       0x0A

class ads1292r
{
  public:
    static void ads1292_Init(void);
    static void ads1292_Reset(void);
     static void ads1292_Reg_Write (unsigned char
READ_WRITE_ADDRESS, unsigned char DATA);
     static void ads1292_Reg_Read (unsigned char
READ_WRITE_ADDRESS);
    static void ads1292_SPI_Command_Data(unsigned char data_in);
    static void ads1292_Disable_Start(void);
    static void ads1292_Enable_Start(void);
    static void ads1292_Hard_Stop (void);
    static void ads1292_Start_Data_Conv_Command (void);
    static void ads1292_Soft_Stop (void);
    static void ads1292_Start_Read_Data_Continuous (void);
    static void ads1292_Stop_Read_Data_Continuous (void);
    static char* ads1292_Read_Data(void);

};

#endif
```

7.2.3. Cuerpo de la librería:

```
////////////////////////////////////////////////////////////////////
////////////////////////
//
//   Arduino Library for ADS1292R Shield/Breakout
//
//   Copyright (c) 2017 ProtoCentral
//
//   This software is licensed under the MIT
License(http://opensource.org/licenses/MIT).
//
//   THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY
KIND, EXPRESS OR IMPLIED, INCLUDING BUT
//   NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR
A PARTICULAR PURPOSE AND NONINFRINGEMENT.
//   IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE
FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY,
//   WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
FROM, OUT OF OR IN CONNECTION WITH THE
//   SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
//
//   Requires g4p_control graphing library for processing.  Built
on V4.1
//   Downloaded from Processing IDE Sketch->Import Library->Add
Library->G4P Install
//
////////////////////////////////////////////////////////////////////
////////////////////////

#include <Arduino.h>
#include <ads1292r.h>
#include <SPI.h>

char* ads1292r::ads1292_Read_Data()
{

   static char SPI_Dummy_Buff[10];

   digitalWrite(ADS1292_CS_PIN, LOW);

      for (int i = 0; i < 9; ++i)
      {
           SPI_Dummy_Buff[i] =
SPI.transfer(CONFIG_SPI_MASTER_DUMMY);

      }

    digitalWrite(ADS1292_CS_PIN, HIGH);

      return SPI_Dummy_Buff;
}

void ads1292r::ads1292_Init()
{
```

```
  // start the SPI library:
  SPI.begin();
  SPI.setBitOrder(MSBFIRST);
  //CPOL = 0, CPHA = 1
  SPI.setDataMode(SPI_MODE1);
  // Selecting 1Mhz clock for SPI
  SPI.setClockDivider(SPI_CLOCK_DIV16);

  ads1292_Reset();
  delay(100);
  ads1292_Disable_Start();
  ads1292_Enable_Start();

  ads1292_Hard_Stop();
  ads1292_Start_Data_Conv_Command();
  ads1292_Soft_Stop();
  delay(50);
  ads1292_Stop_Read_Data_Continuous();
     // SDATAC command
  delay(300);

  ads1292_Reg_Write(ADS1292_REG_CONFIG1, 0x00);       //Set
sampling rate to 125 SPS
  delay(10);
  ads1292_Reg_Write(ADS1292_REG_CONFIG2, 0b10100000); //Lead-off
comp off, test signal disabled
  delay(10);
  ads1292_Reg_Write(ADS1292_REG_LOFF, 0b00010000);         //Lead-
off defaults
  delay(10);
  ads1292_Reg_Write(ADS1292_REG_CH1SET, 0b00000000);  //Ch 1
enabled, gain 6, connected to electrode in
  delay(10);
  ads1292_Reg_Write(ADS1292_REG_CH2SET, 0b00000000);  //Ch 2
enabled, gain 6, connected to electrode in
  delay(10);
  ads1292_Reg_Write(ADS1292_REG_RLDSENS, 0b00101100); //RLD
settings: fmod/16, RLD enabled, RLD inputs from Ch2 only
  delay(10);
  ads1292_Reg_Write(ADS1292_REG_LOFFSENS, 0x00);      //LOFF
settings: all disabled
  delay(10);

           //Skip register 8, LOFF Settings default
  ads1292_Reg_Write(ADS1292_REG_RESP1, 0b11110010);
     //Respiration: MOD/DEMOD turned only, phase 0
  delay(10);
  ads1292_Reg_Write(ADS1292_REG_RESP2, 0b00000011);
     //Respiration: Calib OFF, respiration freq defaults
  delay(10);
  ads1292_Start_Read_Data_Continuous();
  delay(10);
  ads1292_Enable_Start();
}
```

```
void ads1292r::ads1292_Reset()
{
  digitalWrite(ADS1292_PWDN_PIN, HIGH);
  delay(100);                         // Wait 100 mSec
  digitalWrite(ADS1292_PWDN_PIN, LOW);
  delay(100);
  digitalWrite(ADS1292_PWDN_PIN, HIGH);
  delay(100);
}

void ads1292r::ads1292_Disable_Start()
{
  digitalWrite(ADS1292_START_PIN, LOW);
  delay(20);
}

void ads1292r::ads1292_Enable_Start()
{
  digitalWrite(ADS1292_START_PIN, HIGH);
  delay(20);
}

void ads1292r::ads1292_Hard_Stop (void)
{
  digitalWrite(ADS1292_START_PIN, LOW);
  delay(100);
}


void ads1292r::ads1292_Start_Data_Conv_Command (void)
{
  ads1292_SPI_Command_Data(START);                              // Send
0x08 to the ADS1x9x
}

void ads1292r::ads1292_Soft_Stop (void)
{
  ads1292_SPI_Command_Data(STOP);                   // Send 0x0A
to the ADS1x9x
}

void ads1292r::ads1292_Start_Read_Data_Continuous (void)
{
  ads1292_SPI_Command_Data(RDATAC);                             // Send
0x10 to the ADS1x9x
}

void ads1292r::ads1292_Stop_Read_Data_Continuous (void)
{
  ads1292_SPI_Command_Data(SDATAC);                             // Send
0x11 to the ADS1x9x
}
```

```
void ads1292r::ads1292_SPI_Command_Data(unsigned char data_in)
{
  byte data[1];
  //data[0] = data_in;
  digitalWrite(ADS1292_CS_PIN, LOW);
  delay(2);
  digitalWrite(ADS1292_CS_PIN, HIGH);
  delay(2);
  digitalWrite(ADS1292_CS_PIN, LOW);
  delay(2);
  SPI.transfer(data_in);
  delay(2);
  digitalWrite(ADS1292_CS_PIN, HIGH);
}

//Sends a write command to SCP1000
void ads1292r::ads1292_Reg_Write (unsigned char
READ_WRITE_ADDRESS, unsigned char DATA)
{
  switch (READ_WRITE_ADDRESS)
  {
    case 1:
            DATA = DATA & 0x87;
          break;
    case 2:
            DATA = DATA & 0xFB;
          DATA |= 0x80;
          break;
    case 3:
          DATA = DATA & 0xFD;
          DATA |= 0x10;
          break;
    case 7:
          DATA = DATA & 0x3F;
          break;
    case 8:
          DATA = DATA & 0x5F;
          break;
    case 9:
          DATA |= 0x02;
          break;
    case 10:
          DATA = DATA & 0x87;
          DATA |= 0x01;
          break;
    case 11:
          DATA = DATA & 0x0F;
          break;
    default:
          break;
  }

  // now combine the register address and the command into one
byte:
```

```
  byte dataToSend = READ_WRITE_ADDRESS | WREG;

   digitalWrite(ADS1292_CS_PIN, LOW);
   delay(2);
   digitalWrite(ADS1292_CS_PIN, HIGH);
   delay(2);
  // take the chip select low to select the device:
  digitalWrite(ADS1292_CS_PIN, LOW);
  delay(2);
  SPI.transfer(dataToSend); //Send register location
  SPI.transfer(0x00);      //number of register to wr
  SPI.transfer(DATA);      //Send value to record into register

  delay(2);
  // take the chip select high to de-select:
  digitalWrite(ADS1292_CS_PIN, HIGH);
}
```

7.2.4. Declaración de honor